Learning to Retrieve Relevant Passages and Questions in Open Domain and Community Question Answering

Nouha Othman

University of Tunis

Institut Supérieur de Gestion

THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF COMPUTER SCIENCE

Learning to Retrieve Relevant Passages and Questions in Open Domain and Community Question Answering

---

Nouha Othman

defended on the 11th of November 2020

before a jury composed of:

Zied Elouadi	Professor, ISG, University of Tunis	Chair
Lamjed Ben Saïd	Professor, ISG, University of Tunis	Reviewer
Latifa Ben Arfa Rabai	Senior lecturer, ISG, University of Tunis	Reviewer
Kaouther Nouira Ferchichi	Senior lecturer, ISG, University of Tunis	Examiner
Rim Faiz	Professor, IHEC, University of Carthage	Thesis Director
Kamel Smaïli	Professor, LORIA, University of Lorraine	Thesis Co-supervisor

---

2019-2020

Laboratories/ Research units: LARODEC (Tunisia) and LORIA (France)

Abstract

Question Answering (QA) aims to directly return succinct and accurate answers to natural language questions. Passage Retrieval (PR) is deemed to be the kernel of a typical QA system where the goal is to reduce the search space from a huge set of documents to a few number of relevant passages, from which the required answer can be found. Although there has been an abundance of work on this task, it still requires non-trivial endeavor. Recently, community Question Answering (cQA) services have evolved into a popular way of online information seeking, where users can interact and exchange knowledge in the form of questions and answers. The Question Retrieval (QR) problem in cQA is to certain extent analogue to the PR task in traditional QA.

While passage retrieval matches the user question with the document passages to search for correct excerpts in response to the user, question retrieval matches the user's question with the archived questions to find out those that are semantically similar to the queried one. By the time, with the sharp increase of community archives and the accumulation of duplicated questions, the QR problem has become increasingly alarming and it remains more challenging than PR due to the shortness of the community questions as well as the lexical gap problem. In this thesis, we tackle both tasks: PR in open domain QA and QR in cQA. We propose different approaches to improve these critical problems in different languages. For PR, we were mainly based on SVM and n-grams while for QR, we were opted for neural networks mainly word embeddings and Long Short-Term Memory (LSTM). We run our experiments on large scale data sets from CLEF and Yahoo! Answers in different languages to show the efficiency and generality of our proposed approaches. Interestingly, the obtained results transcend that of other previously proposed ones.

keywords:
Question answering, Passage retrieval, Community question answering, Question retrieval, N-grams, SVM, Word embeddings, LSTM

Résumé

Les systèmes de questions-réponses (SQR) visent à retourner automatiquement des réponses concises et précises à des questions posées en langage naturel humain. La recherche des passages (RP) est considérée comme le noyau d'un SQR typique, dont l'objectif est de réduire l'espace de recherche d'un vaste ensemble de documents à un petit nombre de passages pertinents, à partir desquels la réponse requise peut être trouvée. Bien que de nombreux travaux aient été effectués sur cette tâche, des efforts non négligeables restent nécessaires. Récemment, les services communautaires de questions-réponses (cQR) ont évolué pour devenir un moyen populaire de recherche d'informations en ligne, où les utilisateurs peuvent interagir et échanger des connaissances sous forme de questions et de réponses. Le problème de recherche des questions (RQ) dans cQR est dans une certaine mesure, analogue à la tâche d'extraction de passages dans les systèmes de questions-réponses traditionnels.

Tandis que la recherche des passages apparie la question de l'utilisateur avec les passages de documents pour rechercher les extraits corrects en réponse à l'utilisateur, la recherche des questions apparie la question de l'utilisateur aux questions archivées pour trouver celles qui sont sémantiquement similaires à la requête qui a été interrogée. Avec le temps, avec la forte augmentation des archives de la communauté et l'accumulation de questions dupliquées, le problème de RQ est devenu de plus en plus alarmant et il reste plus difficile que RP en raison de la brièveté des questions de la communauté ainsi que le problème du trou lexical. Dans cette thèse, nous abordons les deux tâches: RP dans QR dans le domaine ouvert et RQ dans cQR. Nous proposons différentes approches pour améliorer ces problèmes critiques dans différentes langues. Pour le problème de RP, nous sommes principalement basés sur les SVM et les n-grammes, tandis que pour RQ, nous avons opté pour les réseaux de neurones, principalement les 'word embeddings' et la mémoire à court terme (LSTM). Nous menons nos expériences sur des ensembles de données à grande échelle de CLEF et Yahoo! Réponses dans différentes langues pour montrer l'efficacité et la généralité des approches proposées. Fait intéressant, les résultats obtenus transcendent ceux d'autres proposées précédemment.

Mots-clés:
Questions-réponses, Recherche des passages, les services communautaires de questions-réponses, Recherche des questions, N-grammes, SVM, Word embeddings, LSTM

Acknowledgments

The work presented in this thesis would not have been possible without the encouragement of many people to whom I would like to express my gratitude.

First and foremost, I would like to extend my heartfelt thanks and my deep gratitude to my principal supervisor, Prof. Rim Faiz, not only for teaching me fundamental research principles, but also for her guidance and precious advice, allowing me to improve this work. I am indebted to her for the constant support and encouragement given to me during the past six years. I would also like to give special thanks to Prof. Kamel Smaïli for being my co-supervisor. He taught me a lot about natural language processing and deep learning. He was always there to help me, provided me with valuable support, insightful comments and continuous feedback for my various questions. I'm extremely grateful to have the opportunity to work with him.

I also acknowledge the professors of ISG Tunis and the LARODEC and LORIA members for their crucial and continued assistance.

My warmest thanks are due to my friends and all my family, especially my dear parents Ibrahim and Houda and my beloved sisters Rania and Haifa for their endurance and enormous support. They are the reason behind my successes and the inspiration behind my endeavors. I would like to dedicate this thesis to them.

Finally, it is with immense pleasure and great honor that I acknowledge the jury's members for accepting to evaluate my dissertation.

Contents

List of Figures

List of Tables

List of Algorithms

Glossary

ANN Artificial Neural Networks.

BPTT Propagation Through Time.

CBOW Continuous Bag-Of-Words.

CLEF Cross Language Evaluation Forum.

CNN Convolutional Neural Networks.

CQA Community Question Answering.

cQA community Question Answering.

DR Document Retrieval.

GRU Gated Recurrent Unit.

IE Information Extraction.

IR Information Retrieval.

IRS Information Retrieval System.

LDA Latent Dirichlet Allocation.

LM Language Model.

LSA Latent Semantic Analysis.

LSI Latent Semantic Indexing.

LSTM Long Short-Term Memory.

MRR Mean Reciprocal Rank.

NE Named Entity.

NER Named Entity Recognizer.

NLP Natural Language Processing.

NN Neural Networks.

Okapi BM25 BM stands for Best Matching.

POST Part-of-Speech-Tagging.

PR Passage Retrieval.

QA Question Answering.

QAS Question Answering System.

QLLM Query Likelihood Language Model.

QR Question Retrieval.

RankSVM Ranking Support Vector Machine.

RNN Recurrent Neural Networks.

SVD Singular Value Decomposition.

SVM Support Vector Machine.

TF-IDF Term Frequency–Inverse Document Frequency.

TM Translation Model.

VSM Vector Space Model.

WWW World Wide Web.

Chapter 1

Introduction

1.1 Question Answering context

With the flourishing of Internet, the World Wide Web (WWW) has become an important knowledge repository and an ubiquitous information tool. The overwhelming plethora of information on the Web makes it an attractive resource for people to search for information. Nevertheless, seeking useful and accurate information from such a tremendous repository without efficient tools is just like looking for a needle in a haystack. Hence, with the sharp increase of information, Web search engines such as Google and Yahoo Search have risen as powerful tools to return valuable information in response to a user's query.

Although Web search engines have made great strides over the last years, the problem of effectively searching and locating information on the Internet is still far from being solved. Indeed, traditional search engines mainly deliver a list of potentially relevant documents and web pages instead of directly returning the accurate answer. Thus, it is for the user to formulate good queries, select the appropriate keywords and complete the task of search engines by browsing through the result list to find the desired answer, which is a tedious task and requires not only an additional effort but also a considerable loss of time. Furthermore, most current search engines perform their text search and retrieval using keywords, rather than considering the intent and contextual meaning of natural language queries when serving content to users. Such shortcomings not only hinder the user from getting directly the ultimate answers, but also lead to an overhead of converting search queries into lists of keywords.

Therefore, in recent years, there has been a burgeoning interest in an exciting Information Retrieval (IR)-related area that goes beyond simple document retrieval namely, Question Answering (QA)(Voorhees, 2001). Unlike IR, QA endeavors in directly providing accurate and

succinct answers to user's questions from its knowledge base and it requires advanced Natural Language Processing (NLP) techniques. Indeed, QA needs to cope with several issues and challenges, mainly, the huge collection of documents from which the response should be retrieved, the wide range of question types and how to make a machine understanding the query given in human natural language. A further challenge is the choice of techniques employed to retrieve the relevant answers that are most likely to fit the user's query. Moreover, QA needs to satisfy its domain which can be divided into two categories: closed-domain QA which deals with questions under a specific domain (for example: computer science, biology, medicine (Abacha & Zweigenbaum, 2015), etc.) and open-domain QA which deals with general questions in various domains without any limitation such as in (Fader et al., 2014), (Sun et al., 2015), (Lin et al., 2018), (Izacard & Grave, 2020) and (J. Lee et al., 2020). Several QA Systems (QAS)s are available today and they use various approaches (da Silva et al., 2020) but most of them are language dependent, cover only a specific domain, and often break down for complicated questions. Hence, further work in this area is required to try new approaches and enhance the performance of existing systems.

Despite the fact that there has been a variety of proposed architectures such as (Hirschman & Gaizauskas, 2001), (Tellex et al., 2003) and (Buscaldi et al., 2010), a typical QAS can be thought as a pipeline entailing four principle modules: question analysis, document search, passage retrieval and answer extraction, where each module has to deal with specific issues and different technical challenges.

Firstly, the question analysis determines details about the given question such as its type or class, its focus, and the answer type, which are later used to create the query. Secondly, document retrieval searches from the collection for a set of potentially relevant documents that are most likely to contain the appropriate answer to the question. Then, PR is used to retrieve relevant text snippets called passages from the collection of documents. Finally, answer extraction extracts candidate answers from the retrieved passages, ranks them and returns the highest ranked one as the final answer and likewise the output of the QAS. In some cases, a simple extraction might be not sufficient so, the answer should be formulated or automatically summarized (Chali et al., 2015).

Over the last decade, with the blooming of Web 2.0 and 3.0, a huge amount of user generated content has become an essential information resource on the web, including the traditional Frequently Asked Questions (FAQ) archives and the emerging community question answering (cQA), such as Yahoo! Answers [1], Stackoverflow [2], Quora [3], LinuxQuestions [4] and Live QnA [5].

[1] http://answers.yahoo.com/
[2] http://stackoverflow.com/
[3] https://fr.quora.com/
[4] http://www.linuxquestions.org/
[5] http://qna.live.com/

CQA are defined by Liu et al. (2008) as platforms that enable users to answer to others users' questions, where the content is often structured as questions and lists of corresponding answers associated with metadata such as user chosen categories to questions which are useful for various tasks like QA. Recently, cQA has become a popular and exciting paradigm that allows people with various backgrounds to interact and share information and is deemed to deliver quick and precise answers to complex natural language questions mainly the opinion ones. CQA services has numerous advantages over the traditional ad hoc information retrieval insofar as, instead of returning a full list of web documents, a cQA directly returns short and relevant answers proposed by the community. Additionally, instead of using a set of keywords, the user can express his information need explicitly and clearly by entering a human natural language question that may be previously asked by another user. In cQA, users are a part of an online community where they can contribute by asking questions, giving answers and also voting for the best posts. Unlike traditional QA, cQA provides users with personalized experience and encourages collaboration as well as knowledge sharing among users.

Indeed, cQA services build up very large scale archives of previously asked questions and their corresponding answers. Nonetheless, the information contained within these archives is rarely exploited, this fact is clearly witnessed by the high redundance of questions and answers in online communities. Furthermore, many cQA archives lack semantic information and relevant criteria to browse the archive.

In fact, a user who submits his question may have hundreds of possible answers. Obviously, it will be very time-consuming and disturbing for information seekers to read all the available answers and winnow through them all to find the relevant one. Thus, reducing the lag of time for finding an answer and making full use of the available archives may constitute one of the major challenges. As the archives grow exponentially, the indexing, searching and extracting techniques need to keep high efficiency both in time and retrieval results.

1.2 Motivations and problem statement

With the sharp increase of the cQA archives, numerous duplicated questions have been amassed. Therefore, users cannot easily find the answers they need and consequently post new questions that already exist in the archives. In order to take full advantage of the huge archives of question-answer pairs and reduce the time lag required to get a new answer, it is critical to detect the historical questions that are semantically equivalent to the queried ones. If good matches are detected, the answers to equivalent previous questions can be used to answer the new posted query. This can avoid the lag time caused by waiting for other users to respond, thus improving user satisfaction. The task of retrieving semantically equivalent questions to the new queries from

the cQA archives, known as the Question Retrieval (QR) task, has recently sparked great interest (Xue et al., 2008; X. Cao et al., 2010; Cai et al., 2011; Singh, 2012; K. Zhang et al., 2014; Nakov et al., 2017; Ye et al., 2017). Our thesis falls within this context. We tackle the question retrieval problem by exploring methods to represent the questions in a way that captures semantic, syntactic and context information and measure the semantic similarity between questions.

The QR issue in cQA is to certain extent analogue to the PR task in traditional QA. While PR matches the user's question with the document's passages to search for correct excerpts in response to the user, QR matches the user question with the archived questions to find out those that are semantically similar to the queried one. As a matter of fact, in cQA, the accuracy of the returned questions constitutes a big concern for end users. The QR task remains more tricky than the PR task since the queried question and the archived ones often share very few common words or phrases.

We emphasize that PR was also tackled in this thesis owing to its importance. PR is deemed to be the kernel of a typical QAS where the goal is to reduce the search space from a huge set of documents to a few number of passages. Furthermore, the performance of the entire system significantly depends on the performance of the PR module. Obviously, a QA cannot find the right response to a given question, unless the answer exists in one of the retrieved passages. We study this task in Open domain as the techniques used are not tailored toward a specific domain. Therefore, in this thesis, we tackle two crucial problems: Retrieving passages (PR) in open domain QA and retrieving questions (QR) in cQA.

1.3 Research challenges of retrieving passages and questions

PR and QR impose several research challenges faced in this thesis. Retrieving a relevant passage from a massive document collection is a not trivial task and most existing QASs are still unable to correctly detect the best related text excerpts with regard to the questions from a repository even though it contains the correct passages. The main challenges are the big size and the heterogeneity of the passages which mostly share few common words with the questions. As we focus on open domain QA, the questions and the passages are various and not limited to a specific domain. Another challenge is to understand the meaning of the query and the passages rather than just consider their keywords.

QR remains more challenging than PR as questions in cQA varies significantly in terms of vocabulary, length, style, and content quality. Community questions are unformal, usually affected by noise and verboseness, including greetings, spelling errors and incorrect grammar and punctuation marks. The major challenge is the lexical gap between the queried questions and the existing ones in the archives (Xue et al., 2008), which constitutes a barricade to traditional Infor-

mation Retrieval (IR) models since users can formulate the same question employing different wording. For example, the questions *How can I slow down signs of aging naturally?* and *What are some home remedies to keep your skin looking younger?* have the same meaning but they are lexically different and then may regarded as dissimilar. The word mismatch is indeed a tricky problem in cQA since questions in forums are mostly short and similar ones often have sparse representations with little word overlap. In order to bridge the lexical gap problem in cQA, most previous attempts focus on improving the similarity measure between questions while it is tricky to set a compelling similarity function for sparse and discrete representations of words. Moreover, most existing approaches neither take into account the contextual information nor capture enough semantic relations between words. Language is another big challenge, mainly Arabic as it is a highly inflected and derivational language. Its morphology, ambiguity and its syntactic flexibility have usually been reported as real challenges for most NLP tasks.

1.4 Research questions

From the aforementioned challenges, it is worth asking the following questions that we will try to answer in the current research:

- How to return a correct, precise and concise answer to every question in every language from any corpus?
- How to enhance the performance of existing QASs, increase the number of correct passages and ensure their relevance? This leads to the following subquestions:
  - Is it possible to build a QAS that can automatically give a precise passage to open domain questions in different languages?
  - Can we jointly train a model on PR and combine different features and get better performance on our PR task?
- How to improve the QR task to take advantage of the large scale available archives of community question-answer pairs and reduce the lag of time for delivering an answer? We refer to this us our main objective for our main task. Then, we consider the following subquestions:
  - Can word embeddings offer high-quality vector representations that capture the semantic and syntactic similarity between the community questions?

- Can a neural network based model be successful in retrieving semantically similar questions?
- Is it possible to build a general QR approach that give sufficient precision not only in English but also in fiendishly difficult languages like Arabic?

1.5 Contributions

The main contributions of this thesis can be summarised as follows:

- We propose a novel approach for retrieving and reranking passages according to their relevance. Our approach relies on a new measure of similarity between a passage and a question for the extraction of the candidate passages based on the dependency degree of n-gram words of the question in a given passage. The retrieved passages are re-ranked using a Ranking SVM model combining different text similarity measures including our proposed measure as well as other different semantic and lexical features which have already been proven successful in the Semantic Textual Similarity (STS) task. Our experimental results demonstrate a comparable performance in different languages with other PR competitive approaches.
- We explore the use of word embeddings to capture the semantic similarity of the community questions. We present a word embedding based approach for the QR task in cQA. The similarity between the clustered questions is measured based on their weighted continuous valued vectors. We run our experiments on real world data set harvested from Yahoo! Answers in both English and Arabic to show the efficiency and generality of the proposed approach. As there is no large Arabic dataset available for the QR task, for our experiments in Arabic, we constructed our Arabic dataset by translating the English collection using Google Translation, the most widely used free online machine translation tool, with a careful check of the produced Arabic questions.
- We introduce a deep learning approach based on a Siamese architecture with LSTM networks, augmented with an attention mechanism. This latter is an additional layer, which lets the model give different words different attention while modeling questions. We tested different similarity measures to compare the final hidden states of the LSTM layers. To evaluate the proposed approach, we conducted experiments on large-scale datasets in English and Arabic and the results show that our approach yields significant improvements on QR.

1.6 Publications

Our work on the given issues has given rise to several significant publications:

International journals

- Faiz, R., and Othman, N., Retrieving Relevant Passages using N-grams for Open-Domain Question Answering, International Journal on Artificial Intelligence Tools (IJAIT journal) (impact factor) 28(7): 1950021, 2019.
- Othman, N., Faiz, R., and Smaili, K., Using Word Embeddings to Retrieve Semantically Similar Questions in Community Question Answering, International Science and General Applications (ISGA journal), 2018.
- Othman, N., and Faiz, R.,"A Relevant Passage Retrieval and Re-ranking Approach for Open-Domain Question Answering,", EGC. Revue des Nouvelles Technologies de l'Information vol. RNTI, Ed. Hermann, pages 111-122 , 2016.
- Othman, N., and Faiz, R.,"Question Answering Passage Retrieval and Re-ranking Using N-grams and SVM", Computación y Sistemas, ISSN 1405-5546. (Indexed in: Scopus, Redalyc, E-Journal, e-revist@s, Latindex, DBLP) pages 483–494, 2016.

International conferences

- Othman, N., Faiz, R., and Smaili, K., Manhattan Siamese LSTM for Question Retrieval in Community Question Answering, In proceedings of In OTM Confederated International Conferences "On the Move to Meaningful Internet Systems". Springer, 661-677, Rhodes, Greece, 22-23 October, 2019.
- Othman, N., Faiz, R., and Smaili, K., Enhancing Question Retrieval in Community Question Answering Using Word Embeddings, In proceedings of the 23rd International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES), Elsevier, 485-494, Budapest, Hungary, 4-6 September, 2019.
- Othman, N., Faiz, R., and Smaili K., A Word Embedding based Method for Question Retrieval in Community Question Answering, In proceedings of the International Conference on Natural Language, Signal and Speech Processing (ICNLSSP 2017), Casablanca, Morocco, 05-06 December 2017.

Book chapter

- Othman, N., and Faiz, R.,"A multilingual approach to improve passage retrieval for automatic question answering", In International Conference on Applications of Natural Language to Information Systems, Springer, pages 127–139, 2016.

Other publications

- Menacer, M. A., Abidi, K., Othman, N., Smaïli, K. . Analyse de sentiments des vidéos en dialecte algérien, In Actes de la 6ème conférence conjointe Journées d'Études sur la Parole (JEP-TALN2020), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Volume 2: (pp. 296-304). ATALA, Nancy, France 8-19 Juin, 2020
- Alkhair, M., Meftouh, K., Othman, N., and Smaili, K., An Arabic corpus of fake news: Harvesting, analyzingand classifying, In proceedings of the 7th International Conference on Arabic Language Processing (ICALP2019), Springer, 292-302, Nancy, France 16-17 October, 2019
- Ghorbel, H., Faiz, R., Othman, N., Towards Personalized Keyword Search over Relational Databases, Actes du 37ième Congrès (INFORSID 2019), Hermes-Lavoisier, 187-199, Paris, France, 11-14 Juin, 2019
- Ghorbel, H., Othman, N., and Faiz, R., Recommendation-based Keyword Search over Relational Databases, Revue des Nouvelles Technologies de l'Information (EGC 2018), vol. RNTI, Ed. Hermann, pp.347-352, 2018

1.7 Thesis outline

The rest of this thesis is structured as follows:

- Chapter 2 provides background knowledge on Question Answering and Community Question Answering. We define some fundamental concepts and broadly present traditional Information Retrieval and its limitations. We further overview word vector representations with a focus on word embeddings and Recurrent Neural Networks.
- Chapter 3 gives a literature review on PR in QA and QR in cQA. In particular, we discuss the state-of the-art existing techniques, ranging from the statistical, the lexical, the syntactic, to the semantic approaches.

- Chapter 4 is devoted to the presentation of the approach proposed for improving the PR task for open domain question answering. We will give details of our novel n-gram similarity measure and present the different features used for the passage similarity prediction. A brief introduction on the Ranking-SVM model will be given. We then detail the experiments carried out to validate our proposed approach and we report the main results compared to those obtained by existing solutions in different languages.

- Chapter 5 goes through the investigation of the use of word embeddings on the performance of QR in cQA. We present our proposed approach based on word embeddings and give the experimental process, datasets and evaluation we used. We also discuss the obtained experimental results in English and Arabic.

- Chapter 6 presented our deep learning approach based on a Siamese architecture with LSTM networks, augmented with an attention mechanism. This latter is an additional layer, which lets the model give different words different attention while modeling questions. We tested different similarity measures to compare the final hidden states of the LSTM layers. Our experiments conducted with large-scale data sets provide empirical evidence to validate the application of LSTM to QR in English and Arabic.

- Chapter 7 draws the general conclusions of this research. We recall the findings and research directions studied in this thesis. We highlight its main contributions and discuss further challenges to be tackled in future works.

Chapter 2

Background

2.1 Introduction

As the amount of information available on the web and in companies is continuously increasing, it is becoming more and more difficult to exploit these information resources without resorting to automation. Thus, the need for search systems has become paramount. In this context, Information Retrieval (IR) which is an active and expanding area, has appeared to support the user in the search process to find the information that fits his need.

In addition to search engines which are the first Information Retrieval System (IRS) and the most visible applications in the context of IR on the web, many other IRSs and various research techniques have emerged. Among these systems are the Question Answering Systems (QAS)s and community Question Answering (cQA) platforms.

This background Chapter is organized as follows: In Section 2.2, we begin by broadly presenting the pillars of QA which are IR, IE and NLP. Then, in Section 2.3, we take a closer look on QA: we define QA, give the architecture of a typical QAS and look into the best known state-of-the-art QASs. Section 2.4 introduces cQA. Finally, we overview SVM as well as certain neural network models which are the key elements for our proposed approaches.

2.2 Overview of the pillars of Question Answering

Indeed, QA lies at the intersection of several fields including in particular NLP, IR and IE. Since the input of a QAS is a natural language text whether in a query form or in documents, it is

reasonable to think of exploiting NLP techniques to understand and process this text. Likewise, as the output of the system is a snippet of text extracted from a vast collection of text documents, it is natural to consider the use of the techniques of information retrieval and extraction. In this Section we will define briefly each field while highlighting the main basic concepts.

2.2.1 Natural Language Processing

Natural Language Processing (NLP) is an engineering discipline that is primarily concerned on the interaction between computers and human language. In other words, it refers to the set of methodologies and techniques that allow a computer program to understand human language as it is spoken.

Current NLP research strategies can be classified according to a model that divides the language analysis into several separate levels that can largely be studied independently. It is noteworthy that there is no full agreement on what these levels are exactly or how they are related, but the generally agreed levels of descriptions are six: phonetics, morphology, phonology, syntax, semantics, pragmatics and discourse.

- **Signal processing:** takes as input spoken words and transform them into text (e.g., phonetics, phonology).
- **Morphological analysis:** analyzes individual words into their components and separates non word tokens, such as punctuation, from the words.
- **Syntactic analysis:** deals with the structure or grammar of the text sentences.
- **Semantic analysis:** deals with the meaning assigned to terms in text sentences.
- **Pragmatic analysis:** aims to determine the meaning of terms in sentences according to their context.
- **Discourse integration:** considers that the meaning of a given sentence can depend on the sentences that precede it and may influence the meanings of the sentences that follow it.

In the case of QASs, some NLP techniques are useful for analyzing the question posed by a user in human language and the documents returned by the search module and extract text snippets having similar linguistic features to those derived from the question. These techniques can also be used for analyzing the most relevant text snippets to identify the final answer. Below, we list the most common NLP tasks:

- **Tokenization:** According to the Standford NLP group, tokenization is the process of chopping running text up into pieces called tokens, using a lexical analysis to identify words by recognizing punctuation, spaces, special characters, numbers, etc. For example 2.1:

  Example 2.1. *Tokenizer's input: Question answering systems give users precise responses*
  Tokenizer's output: Question/ answering/ systems/ give/ users/ precise/ responses

  Basically, tokenization is necessary to carry out text processing and any error made at this step may probably induce serious problems at next stages. Although, these tokens are often loosely referred to words or terms separated by a space character, in reality, the issue is more than merely identifying words delimited by punctuation or spaces especially for complex language (e.g., Japanese, Chinese, Hindi). Therefore, a token can be defined as an instance of a sequence of characters in a document, grouped together as a useful semantic unit for text processing that should be not only linguistically significant, but also methodologically useful. Note that a term is a type that is included in the IRS dictionary whereas a type is the class of all tokens containing the same character sequence.

  It is worth nothing that the indexing process for tokenization is often done by an additional plug-in component that broadly exists commercial and open-source such as Open NLP [1].

- **Stemming and lemmatization:** Owing to grammatical reasons, documents often contain different forms of the same word, such as economy, economically and economist. In addition, there are classes of related words having similar meanings, such as democracy, democratic, and democratization. Sometimes, it would be useful for a search for one of these words to return documents that contain another word in the class. Both stemming and lemmatization aim to reduce inflectional forms and derivationally related forms of a word to a common base form. For example 2.2:

  Example 2.2. *am, are, is → be*
  computer, computers, computer's, computer' → computer
  The result of the following mapping of text will look like:
  The boy's cars are different colors → the boy car be differ color

  Nevertheless, the two terms are not the same, rather they differ in their flavor. Stemming aims to generalize the words with the same stem by returning just one base form for them according to their morphological variants and often includes the removal of derivational affixes. This technique is based on algorithms named stemmers using predefined rules for word reduction. For example 2.3:

[1] https://opennlp.apache.org/

Example 2.3. *The words "amuse", "amusement", "amusing" are reduced to "amus".*

Notice that the most popular algorithm for stemming in english is Porter's algorithm (Porter, 1980).

Unlike Stemming, Lemmatization removes inflectional endings properly ensuring that the root word belongs to the language and returns the dictionary form of a word, which is known as the "lemma", using a vocabulary and morphological analysis of words. For instance, for the token "saw", stemming may return just "s", while lemmatization would return either see or saw regarding the token use (verb or noun). To the best of our knowledge, indexing process for either stemming or lemmatization can also be done by an additional plug-in component that exists commercial and open-source.

- **Stop Words Removal:** According to (Wilbur & Sirotkin, 1992), a stop word can be defined as a word that has the same likelihood of occurring in those documents irrelevant to a query as in those relevant to the query. In other terms, stop words refer to extremely common words such as "the", "is", "at" which are likely to be of negligible value in the process of selecting documents matching a user need and thus it is better to filter them out from the vocabulary entirely.

  It is noteworthy that there is no single universal list of stop words. Actually, there are numerous different lists available which vary in size, and in fact not all NLP tools even use such a list. Generally, the strategy for identifying stop words is to sort the terms by the collection's frequency and then take the most frequent ones often hand-filtered for their semantic content relative to the domain of the collection. Among the common stop words, we cite the determiners (e.g., the, a, an, another), the prepositions (e.g., on, in, under, at, for, by) and the coordinating conjunctions (e.g., and, or, nor, but, so, yet).

2.2.2 Information Retrieval

Definition

Information retrieval (Salton, 1968) is the science concerned with the structure, analysis, organization, storage, searching, and dissemination of information. It has been explored extensively since the fifties and in recent years, it has become increasingly important due to the large amount of accessible information available in a widely distributed form such as the World Wide Web (WWW). It can be also defined as the discipline of extracting useful knowledge (usually text) from a large data collection stored in documents, relational databases, or on the WWW in order to satisfy an information need.

An IRS is a software program that is able to locate information stored on computer systems helping the user to find the useful information he needs. More concretely, the purpose of IRSs is to select all relevant information, for text or graphics entered by a user using the search engine, from a large collection of information resources.

By and large, a typical IRS proceeds by query interpretation, document representation, indexing, retrieving and ranking. Obviously, a perfect IRS should be capable to return only relevant documents. But such a system does not exist yet since search statements are often incomplete, sometimes ambiguous and relevance is a subjective issue and mainly depends on the human opinion.

According to Saracevic (1975), relevance is the correspondence in context between a document and a query. Basically, it is considered as the initial criterion for evaluating an IRS. Owing to its importance, it has been widely explored by researchers who admit that the relevance judgment is based on the similarity degree between the query's representation and the document's content retrieved by the search engine. It is noteworthy that many types of relevance have been proposed (Tamine-Lechani & Calabretto, 2008) such as the algorithmic relevance, the topical relevance, the cognitive relevance.

Google is undoubtedly the most popular search engine in the world. This explains why the concept of Information Retrieval has become firmly anchored in people's minds with Google. However, it is far from being perfect, mainly in terms of information access and management, which integrate the storage of information as well as the ability to find information process. A user willing to interact with a system to help him get the information he needs, finds himself confronted with a huge amount of information which is exponentially increasing and makes him unable to find the information he looks for.

Information Retrieval vs. Question Answering

QA is similar to IR in the sense that both have the same goal; supporting the user in the selection process in order to extract from a huge collection of information, relevant answers that fit his query. However, there are two notable differences between IR and QA. The first one is the form of the query. In the case of IRSs, the description of the desired information often contains the words of the documents that interest the user. This is a form of response, while in the QASs, the query does not just contain the searched words but rather it is a full question.

The second difference is the form of the answer provided by the system. In IRSs, the response to a given question is indirect in the form of documents in which the user can find direct answers to his question, while in a QAS, a precise and concise answer is directly returned to the user.

Despite the above differences, both processes are linked to each other for two major reasons (Hirschman & Gaizauskas, 2001): First, IR techniques are applied in the QAS for document search and are further extended to return not only the relevant documents, but also the most relevant passages from the documents. Second, the IR community has developed a methodology for a conscientious evaluation, the best known of which are the yearly campaigns Text Retrieval Conference (TREC)[2] and Cross Language Evaluation Forum (CLEF)[3]. Thanks to this methodology, numerous researches and advancement in the QA field have spawned.

Basic concepts

Retrieving relevant documents in response to a user request is a common task between IR and QA and it lies in two main phases: indexing and matching. We examine these latter below.

Indexing for Information Retrieval Indexing, also known as information analysis, consists in transforming the documents or the query into a set of identifiers and descriptors which represent their content (Salton & McGill, 1986). Thereby, this task aims to detect the most appropriate terms (keywords) that are likely to represent the document's content in a preprocessing step and thus reduce the volume in which the information is searched.

The result of indexing is a list of terms with generally their related weights which point out the degree of representativity. These terms are organized in an index structure, called inverted index. This latter can be viewed as a table that associates words to the documents that contain them which helps to identify all documents in which a word exists. In order to create this inverted index, a set of preliminary tasks should be performed, which involves not only extracting the right and significant descriptors from the documents but also other tasks such as tokenization, stemming, lemmatization and term weighting.

Term weighting is a crucial step in most of the IR indexing processes as it reflects the importance of a word in a document. It aims at enhancing retrieval effectiveness which depends on two main functions: words deemed relevant to the user's query should be retrieved, and words likely to be irrelevant or even extraneous should be eliminated (Salton & Buckley, 1988).

- **Term Frequency:** The term frequency (tf) factor is a local weight that represents the frequency of occurrence of the terms in the document or query text. More concretely, it underlines the number of times that a specific term occurs in a document. In essence, the more frequent a term is, the more important it becomes in the document description. The

[2] http://trec.nist.gov/
[3] http://www.clef-initiative.eu/

term frequency denoted as $tf(t,d)$ of the term t in the document d is often calculated as follows:

$$\text{tf(t,d)} = 1 + \log f(t,d) \tag{2.1}$$

where $f(t,d)$ is the number of occurrences of the term t in the document d.

- **The inverse document frequency:** The inverse document frequency (idf) is a global weight that reflects the importance of a word across all the collection of documents. More precisely, it counts the number of documents in the corpus that contain a term t (Manning et al., 2008). Basically, the more a term is common in all documents, the less it is discriminatory as it does not allow to distinguish between the documents in the research process, whereas a term whose frequency is high whithin a small number of documents is more useful for the selection of documents. Unlike tf, the idf of a rare term is high, while for a frequent term this value is likely to be low. The inverse document frequency denoted as $idf(t)$ of the term t in the corpus is calculated as follows:

$$\text{idf}(t) = \log \frac{N}{n_i} \tag{2.2}$$

where N is the size of the collection (the total number of documents in a corpus) and n_i is the number of documents containing the term i.

- **Term Frequency-Inverse Document Frequency:** Term Frequency-Inverse Document Frequency (tf-idf) combines the above functions to provide a composite weight for each term in each document (Manning et al., 2008). It is given by:

$$tf.idf(t,d) = tf(t,d) \times idf(t) \tag{2.3}$$

This measure assigns to term t a weight in document d that is: greatest when t occurs several times in a few number of documents, lower when the term occurs either fewer times in a document, or in many documents and lowest when the term occurs in almost all documents.

Matching for Information Retrieval Basically, IRSs involve a research process which is closely linked to the indexing process and aims to identify which set of documents are most likely to be relevant or similar in content to the user's query. When the user expresses his need under a query, the system indexes this latter according to the model of corpus' indexing. Then, it looks at the match between the query and the different documents using a similarity measure between them. Obviously, the higher the similarity value is for a document, the more this latter is favored and better it is ranked.

To the best of our knowledge, there are several matching types explored in the literature. Later, we will present a non exhaustive set of matching types and the main existing approaches in the context of PR. Several IR models have been proposed in the past to provide the appropriate matching. In what follows, we address the major three ones.

- **Boolean model:** The Boolean model is the first model of IR and also the most adopted one used by many IRSs up until today. The core idea of this model is based on whether or not the documents include the query terms (Hiemstra, 2009). In essence, boolean logic operators (e.g., AND, OR, NOT) are used to combine query terms and their corresponding sets of documents. For instance, combining terms with the *AND* operator such as the query political AND security will generate the set of documents that are indexed both with the term political and the term security. This model is that it is easy to implement and enables users to control their system by knowing why a document has been retrieved given a query. However, the main shortcoming of this model is that all terms are equally weighted and consequently, it does not provide a ranking of retrieved documents.

- **Vector space model:** The underlying idea of the vector space model, the one which interests us in our search process, is to represent the query and the documents by vectors of keywords embedded in a dimensional Euclidean space where each word is assigned a weight calculated during indexing. For instance, the vector of a document D can be represented as follows:

$$\overrightarrow{D}\begin{pmatrix} t_1 & w_{1,D} \\ t_2 & w_{2,D} \\ .. & .. \\ t_n & w_{n,D} \end{pmatrix}$$

  The search process consists of comparing the vector of the query with those corresponding to the documents and selects the documents having the closest vectors. This comparison is based on the use of similarity measures between the query and the document which are calculated from weights and taking into account only words in common between the query and the document. The values calculated according to the similarity measures are then used to rank the returned documents. Among the similarity measures used, we cite the following which are the most popular ones:

  - the scalar product:

$$sim(d, q) = \sum_{t_i \in d \cap q} ((w(t_i, d) \times w(t_i, q)) \tag{2.4}$$

– the cosine measure:

$$sim(d, q) = \frac{\sum_{t_i \in d \cap q}(w(t_i, d) \times w(t_i, q))}{\sqrt{\sum_{t_i \in d} w(t_i, d)^2 \times \sum_{t_i \in q} w(t_i, q)^2}} \tag{2.5}$$

where $w(t_i, d)$ is the weight of the term i in the document d and $w(t_i, q)$ the weight of the term i in the query q.

- **Probabilistic model:** The core principle of the probabilistic IR model is to estimate the probability that the document D is relevant for the user giving its query Q. The set of relevant documents to the query Q is called R_Q. Thus, the similarity measure is computed as follows:

$$sim(d, q) = \frac{p(d \in R_{Q|d})}{p(d \in \bar{R_Q})} = \frac{p(d \mid d \in R_Q)}{p(d \mid d \in \bar{R_Q})} \tag{2.6}$$

where $p(d \in R_{Q|d})$ is the probability that the document's content description D belongs to the set of documents that are relevant to the query Q.

- **Language model:** A language model (LM) assigns a probability distribution to a sequence of terms, and is typically trained to maximize the likelihood of word input sequences. The LM provides context to distinguish between words and sentences that sound similar but mean different things. Formally, the LM objective is presented as follows:

$$p(t_1, ...t_n) = \prod_i p(t_i \mid t_1, ...t_{i-1}) \tag{2.7}$$

Where t_i is a term with the index i in the given sequence. For word-level prediction a term corresponds to one word or token, whereas for character-level prediction it is one character.

Note that the LM which is based on determining probability based on the count of the sequence of words can be called as N-gram language model. For example, a LM used to determine the probability of a sequence of 2 words is called Bigram language model.

2.2.3 Information Extraction

Information Extraction (Poibeau, 2003) is the technology of automatically extracting a set of precise structured information from text documents and it dates back to the late seventies. Gaizauskas and Wilks (1998) as well define IE as the activity of extracting a structured information source from an unstructured (mainly text) information source. This structured source is then used either for searching or analysis or data-mining techniques, etc. In most cases, this task is based on NLP, computational linguistics techniques and it is also related to IR and requires learning.

Although IE has to deal with the natural language, it does not try to understand the text as a whole but aims to extract text from the relevant elements. The analysis is performed locally, only parts of the text are considered and the type of information to be extracted is known a priori, in contrast to IR which returns information dealing with a subject expressed by a query whose type is not fixed a priori and the analysis covers all the texts from a collection of documents in a similar and independent manner. Moreover, according to Gaizauskas and Wilks (1998), IR retrieves relevant documents from collections whereas IE extracts relevant information from documents. Ultimately, despite some differences, IR and IE can be complementary and give rise to various applications such as QASs.

Basically, extracting information from a given text requires the application of common sub-tasks on the input text, such as Named Entity Recognition (NER) and Part of speech tagging (POST). These latter are defined below:

Categorizing and part of speech tagging:

Part-of-speech tagging, also known as POS tagging or simply POST, refers to the process of classifying words in a text into their linguistic categories called parts of speech and labeling them accordingly based on their definitions as well as their context in order to obtain a set of labels or tags. First automatic taggers have been appeared since 1950 and have been extensively researched over the past few decades. Tagging has proved useful for numerous tasks such as for NLP, IR, IE and machine translations. For american english, the most common POST corpora are the Brown Corpus described in (Ku et al., 1967) and the Penn Treebank corpus (Marcus et al., 1993). Basically, a word may belong to a grammatical class named part of speech according to the part it plays in a sentence. In english, there are eight main parts of speech: noun, pronoun, verb, preposition, determiner, adjective, adverb, conjunction, and interjection. Let's take a simple example 2.4 the tagged sentence below:

Example 2.4. *" Then, we extract the syntactic and semantic features from the passages".*
[('then', adverb), ('we', pronoun), ('extract', verb), ('syntactic', adjective), ('and', conjunction), ('semantic', adjective), ('features', noun) ('from', preposition), ('the', determiner), ('passages', noun)]

Note that the same word can be classified as a noun in a given sentence and as a verb in another one. For example 2.5, in the example below, plants is a noun in the first sentence and a verb in the second one.

Example 2.5. *Plants/N require water and light.* $\Longrightarrow$ *Plants is a noun*
The farmer plants/V the seeds $\Longrightarrow$ *Plants is a verb*

Thus, parts of speech allow to reduce ambiguity in text by indicating the way in which a word is employed. It is worth noting that POST algorithms fall into two main groups: rule-based POST and stochastic tagging.

Named entity recognition:

NER, also called entity chunking or entity identification, is an important subtask of IE that aims to identify and label the atomic elements in running text known as Named Entities belonging to predefined categories such as the names of persons, organizations, locations, dates quantities, monetary values, times and percentages. Let's take the example 2.6 below:

Example 2.6. *In 2010, Steve Jobs announced that Microsoft would be making an investment of 150,000 shares in Apple worth about 150 million dollars.*
The names entities identified in this sentence are produced in an annotated block of text as follows:
In $[2010]_{Time}$, $[SteveJobs]_{Person}$ *announced that* $[Microsoft]_{Organization}$ *would be making an investment of 150,000 shares in* $[Apple]_{Organization}$ *worth about 150 million dollars.*
In this example, a temporal expression token, a two-token person name and two organization names where each one is consisting of one token have been detected and classified.

It is worth noting that today there exist several existing NER systems mainly for english that achieve a high performance.

2.3 Question Answering

In this Section we will take a closer look on QA field; we will define QA and detailed the architecture of a typical QAS. At last, we will review the main existing QASs.

2.3.1 Definitions

According to the definition given by Ferrucci et al. (2009) published in IBM Research Report, QA is an application area of computer science which attempts to build software systems that can provide accurate answers to questions posted by a human in natural language (e.g., english), while QAS is a software system that provides exact answers to natural language questions for a range of topics.

To highlight some advantages made to QASs by merging the above techniques, we will compare QASs with classic IRSs. Firstly, these latter give the user the feeling of having the answer, but in reality these systems return just a list of documents and it is the job of the user to browse, sort and filter expecting to find an answer to his query.

Indeed, IRSs result in a considerable loss of time with no guarantee of finding a correct and adequate answer, while for the QASs the returned answer is the most minimal amount of information that can fit the need of the user. Hence, he is not supposed to filter the output of the system. Therefore, these systems facilitate the task for the user and save him time. Secondly, to query a classic IRS, the user should master its syntax. For example, to query the Google search engine on the web, you need to enter keywords. If you want that they appear together, these keywords should be enclosed in quotes. In the QAS, the user is not supposed to master a specific syntax; he should just enter his question in his natural language. Thirdly, the user of IRSs assumes that he has good tools to find an answer to a question in any domain. However, when he focuses on a particular one, these systems often fail to offer him the precise answer to his question. In contrast, QASs especially those designed to deal with the questions in a particular domain, can handle it well and therefore return satisfactory answers.

Basically, a QAS can be thought as a pipeline composed of four modules: question analysis, document search, passage retrieval and answer extraction. It is noteworthy that this is not the only architecture for a QAS but most existing systems converge toward the same one by further dividing or combining some components. For instance, Buscaldi et al. (2010) groups document search and passage retrieval in one search module while Hirschman and Gaizauskas (2001) introduces an architecture with six modules: question analysis, collection preprocessing, candidate document selection, candidate document analysis, response extraction and result generation.

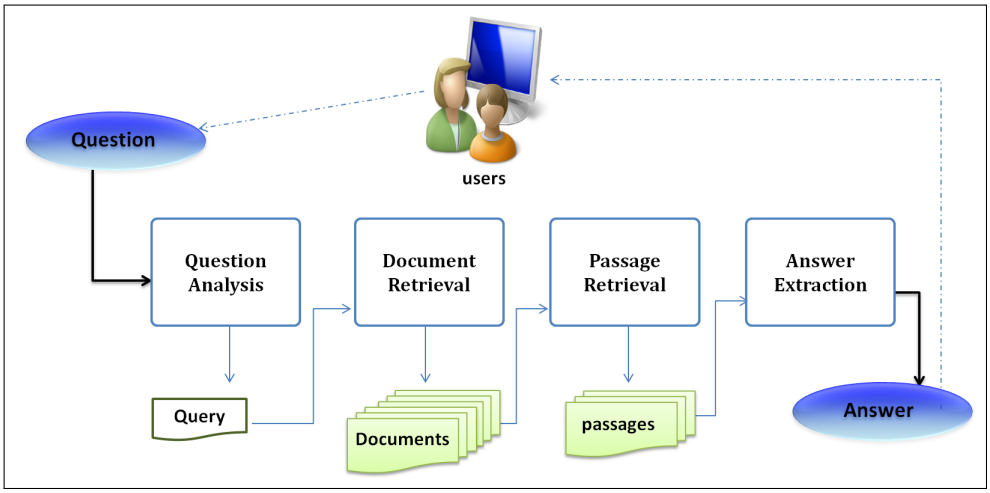


Figure 2.1 – QAS Architecture

In Figure 2.1, we present the architecture of a typical QAS given by Tellex et al. (2003) which is the most popular one and consists of four main modules detailed below.

2.3.2 Architecture

Question analysis

The goal of this module is to process the question given in natural language in order to represent it in a simple form but with more information such as the question class, the answer type, the focus, etc. The output is a query that will be the input of the next module which is the document retrieval. Basically, question analysis has two main tasks:

- Question classification: identifies the type of the input question which is useful for deducing which information is needed to extract the appropriate answer.
- Question formulation: aims to express the main content of the natural language question.

Natural language questions are classified into numerous categories, some of which are simple such as:

- **Factoîd questions:** which require a single fact to be returned as a short answer typically a noun phrase or a simple verb phrase and it is the most widely studied type in QA (e.g., who is Georges Clooney married to?, how old is the current president of Tunisia?)
- **Yes/ No questions:** which just require yes or no as answer (e.g., is Venezuela the most dangerous country in the world?)
- **Definition questions:** which require a full definition as answer (e.g., what is NLP?)
- **List questions:** which require multiple facts as answer (e.g., name the different modules of a typical QAS, list the American States).

Unlike simple questions, complex questions whose answers are expressed by long and articulated sentences or even paragraphs, often require multiple different types of information and do not suppose that a single answer can fit all the information needs. Generally, such questions require inferencing and synthesizing information from several documents. Among the common types of complex questions we can cite the following ones:

- **Cause/ consequence questions:** which require causes or consequences as answer (e.g., what are the causes of the Tunisian revolution ?

- **Evaluative or comparative questions:** (e.g., what are the differences between the IRS and the QAS ?)
- **Procedural question:** (e.g., which are the steps to create a website?)
- **Questions about opinion:** (e.g., what do people think about the tunisian revolution?).

Document retrieval

The goal of document retrieval module is to reduce the search space for finding an answer from a huge collection to a smaller set of documents in order to perform deeper analysis. This module returns a ranked list of documents considered most likely to contain the answer to the question extracted from a corpus.

Basically, the documents are the initial source from which the answers are extracted. This source is not limited only to documents but it also involves web pages, databases, knowledge bases, etc. The type of the information source in the QAS influences the choice of approaches and techniques to be used. Obviously, the choice of the information source is a key challenge and it is influenced by the domain, the objective of the system and the resources availability.

Passage retrieval

Passage Retrieval (PR) is a crucial intermediary between document retrieval and answer extraction. It aims to further reduce the search space from a collection of documents to a fixed number of passages. As pointed out in (Callan, 1994), passages are portions of a document text and can be grouped into three types: discourse, semantic and window.

The first type is based on textual units such as sentences, paragraphs, and sections. Semantic passages are based on the content or the subject of the text while window passages are based on a sequence of words. In this context, the key challenge is the choice of the passage type. In addition, a passage should be neither too short nor too long. Indeed, short excerpts may not contain the answer, and long ones may include additional information that can skew the response or require more robust extraction modules. For instance, we propose the following examples 2.7 and 2.8 derived from the CLEF2010 dataset.

Example 2.7. *Question: What percentage of people in Italy relies on television for information?* $\Longrightarrow$ *Passage: In Italy, 80% of the people get their daily information from television. If that television is not broadcasting all voices, then people do not get the chance to make their own decisions. That is fundamental to democracy.*

Example 2.8. *Question: Why was Perwiz Kambakhsh sentenced to death?*
$\Longrightarrow$ *Passage: whereas the 23 year-old Afghan journalist Perwiz Kambakhsh was sentenced to death for circulating an article about women's rights under Islam, and whereas, after strong international protests, that sentence was commuted to 20 years imprisonment.*

Answer extraction

This module returns the final answer to the question posted by the user after filtering the retrieved passages. It often involves a ranking step which aims to order the candidate passages returned by the PR module such that the most relevant ones appear first. Note that answer extraction also referred to as answer selection depends on several factors, among others: the question complexity and the answer type provided by the question processing module. In some cases, a simple extraction is not sufficient so, the answer should be well formulated or summarized. For instance, we can take the same questions associated with the corresponding precise answers as shown in the examples 2.9 and 2.10 below.

Example 2.9. *Question: What percentage of people in Italy relies on television for information?*
$\Longrightarrow$ *Answer: 80%*

Example 2.10. *Question: Why was Perwiz Kambakhsh sentenced to death?*
$\Longrightarrow$ *Answer: for circulating an article about women's rights under Islam*

2.3.3 Evolution of Question Answering systems

Earlier, QASs were mostly domain specific. The first QAS BASEBALL goes back to 1961 (Green Jr et al., 1961). It answered english questions about the baseball games in the american league for one year. LUNAR (Woods, 1973) as well is another well-remembered early QAS. It was designed to answer questions about the chemical analyses of lunar rocks and soil gathered by Apollo 11. Then, Spoken dialog systems appear to enable users to ask them verbally in natural language. For instance, GUS (Bobrow et al., 1977) was a dialog system for airline reservation using structured database as the knowledge source. However, these systems were restricted domains and could only deal with structured data.

With the emergence of the Web, QASs likewise became open domain. START (Katz & Lin, 2003) the world's first Web-based QAS, has been developed at the MIT Computer Science and Artificial Intelligence Laboratory. In response to English questions, it gives users all over the world access to multi-media information using a new knowledge annotation approach.

Today, there are a multitude of different QASs being used. One of the most popular ones is

IBM's Watson (Ferrucci et al., 2010). It is an open-domain QAS, designed to participate to the well-known american TV quiz show Jeopardy and actually starts to be used commercially.

Siri (Lohr, 2012), an intelligent mobile personal assistant from Apple, is another famous commercial spoken QAS giving answers to questions about weather reports, restaurant suggestions, etc. IBM's Watson and Siri have been implemented in healthcare, marketing, finance and education.

On the web, Ask.com is one of the most used QASs searching the internet to provide answers to english questions. It has served millions users within its 15 years of existence. However, most existing QASs cover only a specific domain and are not able to answer complicated questions. Added to that, they are mostly language dependent.

2.4 Community Question Answering

Over the last years, the Internet has changed the way people communicate and exchange. CQA illustrates this change as it emerges as a popular service for users to ask and answer various questions, interact with each other, and access the previous question-answer pairs for the fulfillment of different information needs. The examples of such community services include Yahoo! Answers [4], Stackoverflow [5], MathOverflow [6], Quora [7], WikiAnswers [8], LinuxQuestions[9], Live QnA [10], Baidu Zhidao [11], and Google Ejabat [12].

Certain cQAs are general-purpose QA sites such as Yahoo! Answers, the forerunner of cQA, which accepts open domain questions as long as they do not violate the site's guidelines. Focused cQAs have also supported questions and answers that belong to specific subjects such as programming, mathematics, sport and music. For instance, Stack Overflow, is a popular community platform that attracts millions of programmers worldwide and allows them to show their expertise through knowledge sharing activities. Numerous programmers are indeed hired owing to their important contribution to such sites.

MathOverflow serves as a stack exchange web site for mathematicians allowing them to post questions, give answers, and rate both, while earning merit points for their activities.

[4] http://answers.yahoo.com/
[5] http://stackoverflow.com/
[6] https://mathoverflow.net/
[7] https://fr.quora.com/
[8] wiki.answers.com/
[9] http://www.linuxquestions.org/
[10] http://qna.live.com/
[11] zhidao.baidu.com
[12] https://ejaaba.com/

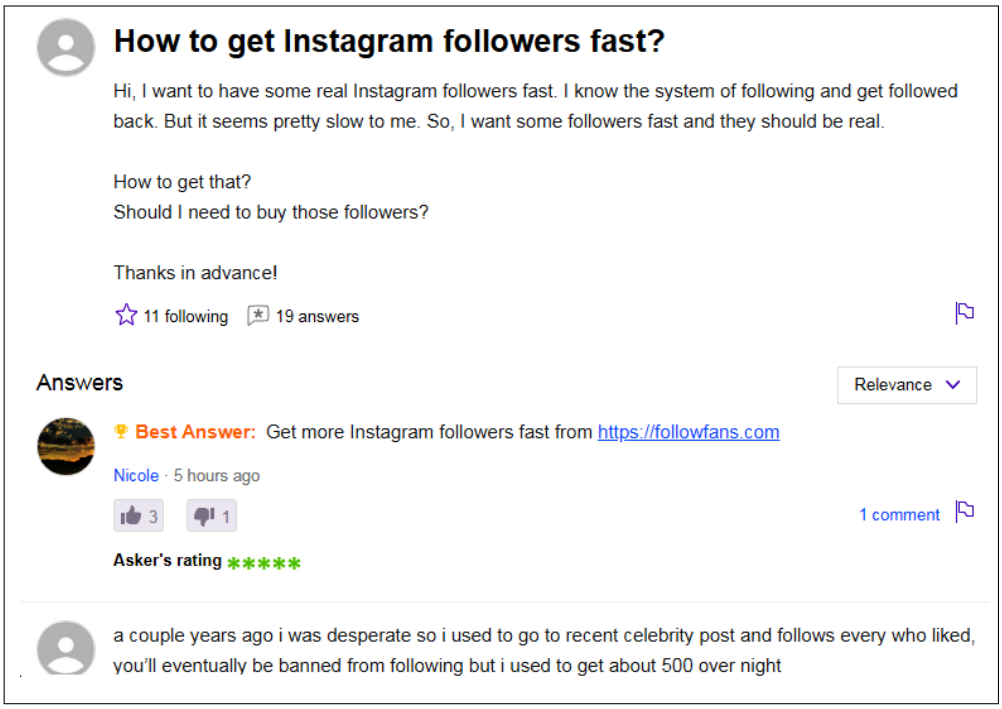


Figure 2.2 – An example of a question thread extracted from Yahoo! Answers

CQA platforms are considered to be automatic systems that take advantage of the collective intelligence or Wisdom of the Crowd by allowing users in an online community to post questions, provide answers and also vote for the existing posts and then collectively garner knowledge and fulfill different information needs. Unlike traditional QA, cQA offers users personalized experiences within a community and encourages collaboration and knowledge sharing among users.

Figure 2.2 shows a snapshot of a question thread discussed in Yahoo! Answers, the most popular cQA, where some key elements are presented such as the posted query, the best answer, the user voting and the user ratings.

In major cQA systems, we have three main components (Shah & Pomerantz, 2010): a mechanism which enables users to submit their natural language questions which can be either a factoîd or complex, a second mechanism allowing users to submit their answers and a platform to make easy the exchange and the interaction between the users. Basically, there exist four major differences between traditional QASs and CQA services (Blooma et al., 2011):

The former difference concerns the question type. In fact, QASs process single-sentence questions which are particularly fact questions, while cQA services handle multiple-sentence questions which are composed of two or more sentences. Thus, constructing a QAS over cQA archives that can handle multiple-sentence questions remains a tricky and challenging task to

better fit user needs. The second difference is related to the answer source. In fact, while questions and answers in a cQA service are generated by ordinary users, the producers of automatic QA content are mainly experts, publishers or journalists. Therefore, they differ in terms of the content size, structure and vocabulary. The third dissimilarity has to do with the answer quality, which obviously reflects the quality of the system. Indeed, in cQA services, answers come from different types of users, with varying knowledge. The quality of these responses depends on the community that participates and becomes more important when there are several answers to a given question. The fourth difference concerns the metadata availability. Unlike traditional QASs, cQA services are rich in metadata such as comments, ratings and authorship. Most of the studies on cQA services exploited these metadata to ensure high-quality content.

Over times, a huge amount of historical question-answer pairs have been amassed. The community archives are continuously increasing accumulating duplicated questions. With such a tremendous amount of data in cQA, users may directly search for relevant previous questions from the available archives instead of looking through a full list of documents from the Web. Hence, the corresponding answers to the matched historical questions can be explicitly retrieved and returned to the user. Therefore, the passage retrieval and answer extraction modules of traditional QASs can be respectively simplified into question retrieval and answer retrieval modules in cQA.

The popularity of the cQA services motivates research in this area in order to exploit the information contained in the community archives by tackling several problems, such as question retrieval, answer quality evaluation and expert users detection.

There has been a host of work on the problem of question retrieval (Xue et al., 2008; X. Cao et al., 2010; Cai et al., 2011; Singh, 2012; K. Zhang et al., 2014; Nakov et al., 2017; Ye et al., 2017; Rücklé, Swarnkar, & Gurevych, 2019; Bae & Ko, 2019) to detect the historical questions that are semantically equivalent to the new queried ones, in order to reduce the time lag required to get a new answer, thus improving user satisfaction. If a similar question is detected, its related answer can be returned as a relevant response to the new posted query.

The task of evaluating the quality of answers in cQA sites has also been subject of wide interest (Shah & Pomerantz, 2010). Unlike traditional QA, cQA builds on a rich history with a huge amount of available metadata which is indicative to finding relevant and high-quality content.

There is also a sub-branch of research using existing best answers to predict certain user behaviors and detect active expert users (Riahi et al., 2012) in order to direct new questions to the right group of experts and therefore, new matching questions will no more be missed and receive a proper answer.

2.5 Support Vector Machine

Support Vector Machine (SVM) was first introduced in (Cortes & Vapnik, 1995) as a supervised machine learning method that can be applied for a discriminative regression or classification, where these latter are performed by determining an optimal separating hyperplane that maximizes the margin named w between two different classes. More concretely, the basic idea behind SVM is to output an optimal hyperplane which can categorize new examples according to labeled training data. Suppose we have a set of training $\{(x_i, y_i)\}_{1 \leq i \leq n}$, where $x_i \in \mathrm{R}^n$ is an N-dimensional vector and $y_i \in \{+1, -1\}$ denotes the class label of i-th instance. A simple example of a separable problem in a two-dimensional space is illustrated in Figure 2.3, where the support vectors (a subset of training samples) are surrounded by dotted blue circles defining the margin of largest separation between the two classes.

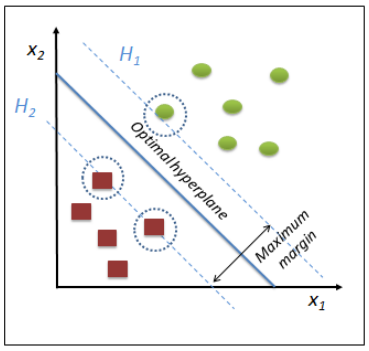


Figure 2.3 – An illustrative example of a separable problem in a 2 dimensional space

Basically, a hyperplane H is formally introduced as follows:

$$w^T x + b = 0 \tag{2.8}$$

where b is a parameter called threshold or biais which represents the intercept for the hyperplane separation. The support vectors are input vectors for which:

$$\begin{aligned} & w^T x + b \geq 1 \quad \text{if } y_i = 1 \text{ (H1)} \\ & or \\ & w^T x + b \leq -1 \quad \text{if } y_i = \text{-1 (H2)} \end{aligned} \tag{2.9}$$

From this, we can deduce a generalized equation of an optimal hyperplane which is the following: $y_i(w^T x_i + b) \geq 1$. Given the distance between H and $H1$ is:

$$w^T x + b = 1, \tag{2.10}$$

the distance between $H1$ and $H2$ is set to: $\left|w^T x + b\right| / \|w\| = 1/\|w\|$. So, the distance between $H1$ and $H2$ which represents the margin between positive and negative examples is: $\frac{2}{\|w\|}$

Recall that in order to define an optimal hyperplane, we need to maximize the width of this margin (w). That is to say, we tend to minimize $\frac{\|w\|}{2}$. Thus, given a variable $n_i \geq 0$ which is introduced for misclassification errors, the optimization problem can be defined as follows:

$$\begin{aligned} &\min_{w,b,n} : \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{1} n_i \\ &s.t. : y_i(w^T x_i + b) \geq 1 \end{aligned} \tag{2.11}$$

where in equation 2.11, the first term denotes the width of the margin while the second term specifies the cost of the misclassification. The decision function $f(x)$ can be set to:

$$f(x) = sign(g(x)) \tag{2.12}$$

where

$$g(x) = \sum_{i}^{n} \alpha_i y_i (x_i.x) + b \tag{2.13}$$

where α_i denotes the weight of training example x_i, ($\alpha_i \geq 0$). Note that the latter equation 2.10 can be rewritten using a kernel function as follows:

$$g(x) = \sum_{i}^{n} \alpha_i y_i k(x_i.x) + b \tag{2.14}$$

Recall that the kernel k is a mathematical mapping function used to map the data into higher dimensional spaces in expectation that in this new higher-dimensional feature space, the data might become easily separated or better structured. It is worth noting that thanks to kernel methods, we can have infinite-dimensional spaces as there is no constraints on the form of this mapping.

According to (Suzuki et al., 2002), SVM provides some advantages over other learning algorithms such as decision trees and maximum entropy method since it offers a high generalization performance even with high dimension feature vectors. Additionally, unlike other learning methods it has a few parameters to set and can also manage kernel functions without increasing computational complexity.

2.6 Word Embeddings

Word embeddings, also known as distributed representations of words refer to a set of machine learning algorithms to build continuous word vectors based on their contexts in a sizeable corpus

using shallow neural networks. They learn a low-dimensional vector for each vocabulary term, where the similarity between the word vectors can capture the semantic and syntactic similarities between the corresponding words. Word embeddings also acquire more complex relationships like gender, tense, geography. A typical example for the analogies that can be discovered using vector arithmetics: $king - man + woman = queen$. Interestingly, the generated vectors can be manipulated arithmetically just like any other numerical vector.

Global Vectors (GLOVE) is a powerful word embedding model proposed by (Pennington et al., 2014). It relies on constructing a global co-occurrence matrix of words in the corpus, where the embedding vectors are based on the analysis of co-occurrences of words in a window. It is based on two main steps. The former one is the construction of a co-occurrence matrix from a training corpus. The second step is the factorization of the constructed matrix in order to get vectors.

Word2Vec, proposed by (Mikolov, Chen, et al., 2013), is the most popular model to learn word embeddings using shallow neural network. It has proven in (Naili et al., 2017) that Word2vec can outperform GLOVE with sizeable text collection in different languages. Basically, word2vec aims to train a model on the context on each word, so similar words will have similar numerical representations using either of two model architectures namely, Continuous Bag-of-Words model (CBOW) and Skip-gram. The former one is trained to predict a current word given its context, while the second does the inverse predicting the contextual words given a pivot word in a sliding window.

2.6.1 CBOW model

The CBOW model predicts the center word given the vector representation of its surrounding words using continuous bag-of-words representation of the context, hence the name CBOW. For example, for the context *guy*, *attempt*, *over*, *puddle*, *fall*, CBOW is able to predict from these words, the center word *jump*. Figure 2.4, illustrates the CBOW principle which includes three layers according to the data process, namely input layer, project layer and output layer.

- The input layer represents the context as a bag of words.
- The hidden layer corresponds to the projection of the input words into the weight matrix of the model which contains all the words of the vocabulary.
- The output layer receives the weighted sum of inputs and performs the Softmax calculations to output probabilities for the target words from the vocabulary.

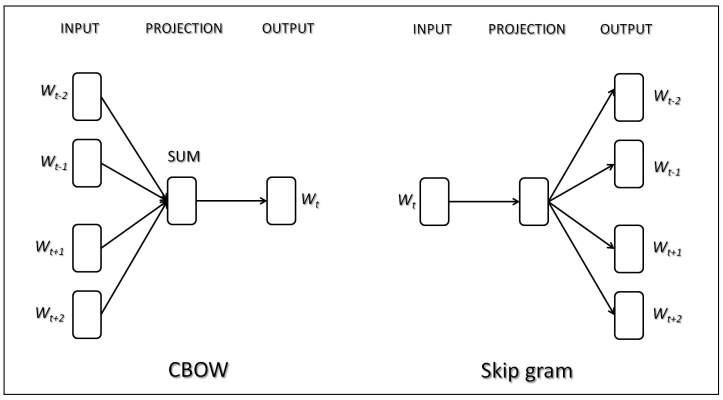


Figure 2.4 – Continuous bag-of-word and Skip-gram models (Mikolov, Chen, et al., 2013)

CBOW works as follows: Each word in the context is projected into the model weight matrix which will be projected on the output layer. Then, the model compares its output with and each word of the context in order to correct its representation based on the back propagation of the error gradient. The latter consists in correcting errors according to the importance of elements that causes these errors such as the model weights. The goal of CBOW is to find the probability of a word occurring in a context. Let consider a corpus with a sequence of words $w_1, w_1, \dots w_T$. The window is defined by parameter c, where c words at the right and left of our target word. The context vectors are summed and used to predict the target. Formally, CBOW aims to maximize the following objective function formula:

$$\frac{1}{T}\sum_{t=1}^{T} logP(w_t \mid \sum_{-c\leq j\leq c, j\neq 0} w_{t+j}) \tag{2.15}$$

2.6.2 Skip gram model

Skip gram does the inverse of CBOW as shown in Figure 2.4. For a given word, it predicts the context from which it is derived based on the three layers: input layer, hidden layer and output layer. The input layer of this network is presented by one single vector that corresponds to a single word instead of a bag of words like in CBOW. In the input layer, the network is fed as a one hot input vector. It then produces a hidden state which is in turn transformed into probabilities using Softmax. For Skip gram, each context is predicted independently given the target. Given a sequence of training words $w_1, w_1, \dots w_T$, the objective of the Skip-gram model is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-c\leq j\leq c, j\neq 0} logP(w_{t+j} \mid w_t) \tag{2.16}$$

The probability is defined as a Softmax, where u_w is the target embedding vector for w and v_w is a context embedding vector. The softmax is used to obtain the posterior distributions of words, which is a multinomial distribution. The following softmax definition is employed for skip-gram, while for CBOW the target and context vectors should be swapped:

$$p(w_c \mid w_t) = \frac{exp(v_{w_c}^T u_{w_t)})}{\sum_{w=1}^{W} exp(v_w^T u_{w_t})} \tag{2.17}$$

Nevertheless, Softmax is expensive to use as a loss function, since computing the gradient has a complexity proportional to the vocabulary size W. In order to overcome this problem, Mikolov, Sutskever, et al. (2013) proposed two solutions namely, Negative sampling (NEG) and Hierarchical softmax (H-Softmax), which is a $O(\log_2 W)$ algorithm for estimating Softmax. The objective of these two functions is to optimize the calculation of the output vectors. The basic intuition behind negative sampling is to update only a sample of the set of output vectors, while Hierarchical softmax is based on the Huffman coding which is based on a lossless data compression algorithm.

2.6.3 CBOW vs. Skip gram

According to (Mikolov, Chen, et al., 2013), each one of these models has its own advantages. CBOW does not rise substantially when we increase the window. Additionally, it allows for a better modeling of frequent words, while Skip gram allows to have a better modeling of infrequent words. Interestingly, Skip gram allows to better capture semantic relationships in small learning corpora while CBOW performs better with sizeable data than Skip-gram.

2.7 Variants of Neural Networks

Artificial Neural Networks (ANN) have been developed to simulate the human brain. A Neural Network "is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use", (Haykin, 1994). The aim of a neural network is to learn to recognize patterns in data.

The neural networks are composed of multiple artificial processing units, called Neurons. As shown in Figure 2.5, a neuron receives a number of inputs, sums up the weighted values and uses an activation function to output a numeric value.

Formally, a neuron is represented by the function f_n, where n is a specific neuron, φ denotes an activation function, $\vec{w}$ is a vector of weights leading to the neuron and $\vec{x}$ is a vector of input

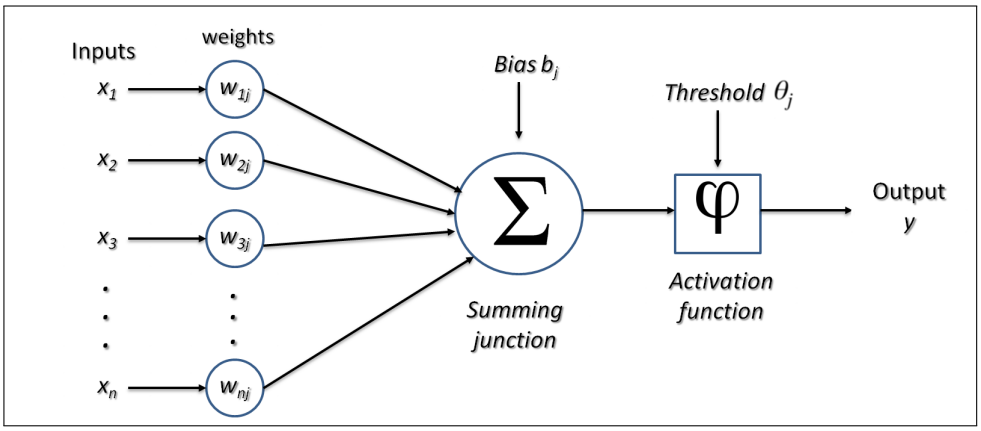


Figure 2.5 – A model of an artificial neuron (Haykin, 1994)

values.

$$f_n = \varphi(\sum_{i=1}^{n} x_i w_{ij}) \tag{2.18}$$

When multiple neurons receive the same inputs, they form a Single Layer Perceptron (SLP) like in Figure 2.5. If neurons in a layer send their outputs to other neurons, we say that multiple layers form a feed-forward neural network, called a Multi Layer Perceptron (MLP) as shown in Figure 2.6. So, an SLP consists only of an input and an output Layer, while an MLP has also a number of hidden layers in between.

The activation function (such as a log function, a sigmoid function, or a hyperbolic tangent) is used to introduce non-linearities in the network by transforming linear combinations into non-linear ones and solve hard problems. The training of neural networks aims to determine a good estimation of parameters (weights) for maximizing a neural network's accuracy. To perform this learning process, a method named Backpropagation is employed.

Backpropagation is based on a gradient descent [13] optimization algorithm (Hochreiter, 1998) to adjust the weight of neurons by calculating the gradient of the loss function where the goal is to minimize that loss. The loss function, composing of several parameters (weights), has a global minimum point as well as multiple local minima. The backpropagation algorithm attempts to determine the global minimum of the loss function, by iterating the following steps:

- Perform a forward pass in the network: feed the network with a training instance and calculate all neurons' output by passing neurons' output to their successive neurons.
- Compute the loss signal and propagate it backwards in the network, by updating all the weights in order to decrease the error.

[13] Gradient descent is an optimization algorithm that uses an iterative process to minimize a given cost function.

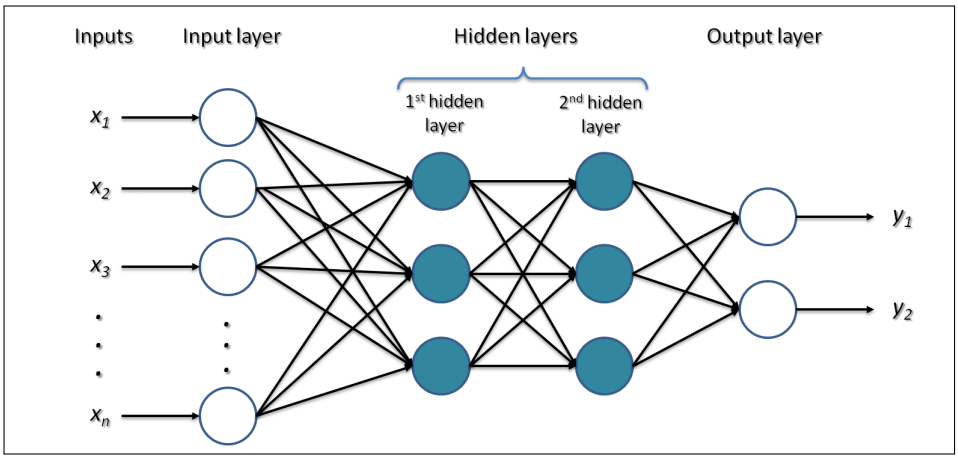


Figure 2.6 – An example of a multi layer perceptron

Many variants and adaptations of ANNs have arisen over the years, with widely varying properties such as RNN and LSTM.

2.7.1 Recurrent Neural Networks

Unlike feedforward NNs which accept only a fixed size input and output and have no sense of state to consider previous events, RNN can operate over sequential input and produce sequential output. Interestingly, RNN can handle a variable-length sequence input owing to a recurrent hidden state whose activation at each time depends on that of the previous time. Figure 2.7 illustrates what a simple RNN looks like.

The Figure 2.7 shows a vanilla RNN being unrolled into a full network. For instance, if the input sequence is a sequence of 4 words, the network would be unrolled into a 4-layer neural network; one layer for each word.

x_t is the input at time step t. For instance, x_i could be a one-hot vector corresponding to i-th word of a text sequence.

h_t denotes the hidden state and also the memory of the network at time step t. h_t is computed based on the previous hidden state and the input at the current step.

The updates to a hidden-state vector h_t are calculated via:

$$h_t = \sigma(Ux_t + Wh_{t-1}) \tag{2.19}$$

$$o_t = softmax(Vh_t) \tag{2.20}$$

Where σ is a sigmoid function, W is an input to hidden weight matrix, U and V represent the

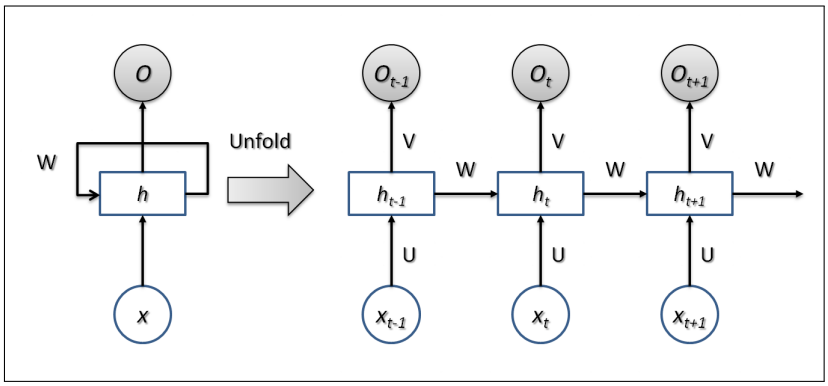


Figure 2.7 – An unrolled recurrent neural network (Vinyals et al., 2015)

weight of neurons and o_t is the output at time step t. For example, to predict the next word in a sentence, it would be a vector of probabilities across the vocabulary.

Similar to traditional NNs, RNN uses a backpropagation algorithm for the training. However, In RNN, the weight of neurons is shared by all time steps in the network and then the gradient at each output depends on both the current and previous time step calculations. For instance, in order to calculate the gradient at time $t = 6$, we have to backpropagate 5 steps and sum up the gradients. This process is known as Back Propagation Through Time (BPTT).

In RNN, the weight of neurons parameters are shared by all time steps in the network and so the gradient at each output depends on the current and previous time step calculations.

Unfortunately, Vanilla RNNs usually have a serious problem when it comes to capture long-term dependencies while training these networks because the gradients tend to either vanish or explode. This is known as the vanishing gradient problem (Bengio et al., 1994). This problem arises due to the product of the same weights several times during back-propagation. In order to solve the vanishing gradient problem, a variant of RNNs with gated cells called LSTM, has been proposed.

2.7.2 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a variant of RNN proposed by (Hochreiter & Schmidhuber, 1997) to solve the vanishing gradient problem faced while training vanilla RNNs with BPTT, when capturing long-distance dependencies in sequential data. LSTM resolves the BPTT problem by ensuring that a constant error is maintained to make the RNN learn over long time steps. LSTM achieves this owing to its gated cells which are integrated to reduce the multiplication of the gradient problem, and make the RNN more efficient in long-term memory tasks. Figure 2.8

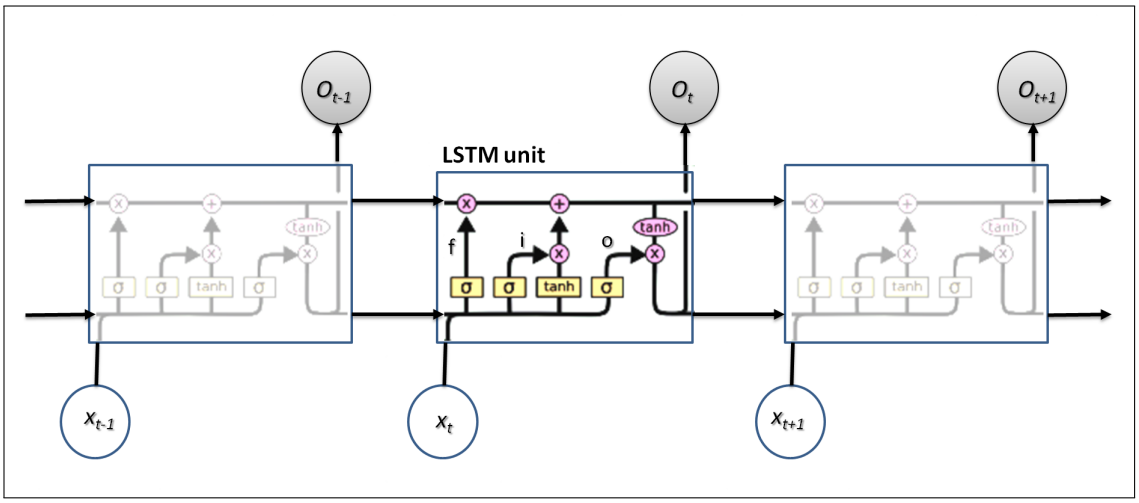


Figure 2.8 – An LSTM cell architecture (Hochreiter & Schmidhuber, 1997)

shows that an LSTM architecture is based on four main components namely, the memory cell (c), the input gate (i), the output gate (o), and the forget gate (f).

The cell states and the different gates constitute the core of LSTM. A cell state performs as a memory to store, read, and remove information according to the decisions made by the input, output, and forget gates that open or close, and each memory cell corresponds to a time-step. The gates regulate the information flow and pass the information based on the weights which are trained by a recurrent learning process.

Given input vector x_t, hidden sate h_t and memory state c_t, the updates in LSTM are performed as follows:

$$i_t = sigmoid(W_i x_t + U_i h_{t-1} + b_i) \tag{2.21}$$

$$f_t = sigmoid(W_f x_t + U_f h_{t-1} + b_f) \tag{2.22}$$

$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2.23}$$

$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \tag{2.24}$$

$$o_t = sigmoid(W_o x_t + U_0 h_{t-1} + b_0) \tag{2.25}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.26}$$

where i_t, f_t, o_t are input, forget, and output gates at time t, respectively. W_k, U_k are LSTM parameterized weight matrices, b_k represents the bias vector for each k in $\{i, f, c, o\}$ and ($\odot$)

represents an element-wise product of matrices, known as the Hadamard product which is simply an entrywise multiplication. Unlike the common matrix product, the Hadamard product is a binary operation that takes two matrices of the same dimensions and returns another matrix of the same dimension, where each element i, j is the product of elements i and j of the original two operands as shown in the example 2.11.

Example 2.11. *The Hadamard product for a* 3×3 *matrix A with a* 3×3 *matrix B is a* 3×3 *matrix as follows:*

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & a_{13}b_{13} \\ a_{21}b_{21} & a_{22}b_{22} & a_{23}b_{23} \\ a_{31}b_{31} & a_{32}b_{32} & a_{332}b_{33} \end{bmatrix}$$

where the Hadamard product is associative, commutative and distributive over addition.

2.8 Conclusion

QA lies at the intersection of several fields including in particular NLP, IR and IE. Indeed, it exploits NLP techniques to understand and process the question posted by the user in natural language. Then, it resorts to the techniques of IR and IE in order to search and extract a brief snippet of text from a huge document collection and return it as an output of the system answering the user's question. QA has attracted so much attention and has been subject of an abundant work since the appearance of the concept. Although, there has been a variety of proposed architectures, a typical QAS can be thought as a pipeline entailing four principle modules: question analysis, document search, passage retrieval and answer extraction where each module has specific challenges to deal with and has also been extensively explored by researchers in order to reach higher performance, speed and accuracy. From our study, we have deduced that PR is the key component in a typical QAS, thus we will focus on this latter to enhance the performance of existing QASs. Likewise, question retrieval is the core of cQAs.

In this Chapter, we have presented QA as well as cQA elucidating their basic concepts and the key related fields. We have also given the general architecture of a typical QAS, detailed its different modules, and cited some examples of well known QASs. We also shed light on the emerging cQA and mentioned the most well known community platforms as well as the main related challenges.

In the next Chapter, we will give an overview of main related works on passage retrieval and question retrieval and highlight their limitations.

Chapter 3

Related work on passage retrieval and question retrieval

3.1 Introduction

Passage retrieval (PR) is considered as one of the key components in a typical QAS and has been widely studied over the years. Similarly, question retrieval (QR) is a core problem in cQA and remains more challenging than PR due to the shortness of the community questions and the lexical gap problem. The problem of finding semantically similar questions is not a new research area and it is closely related to PR. By the time, with the sharp increase of community archives and the accumulation of duplicated questions, this problem became a real challenge. Hence in this chapter, we discuss the main work done till date in the crucial tasks of PR in QA and QR in cQA.

3.2 Related work on passage retrieval and ranking in QA

Existing works on PR use and combine different approaches, where the aim is to reduce the search space from a huge collection of documents to a fixed number of relevant passages and improve the performance of QASs . Basically, any QAS may virtually consist of two main high-level components: retrieval and selection (Echihabi & Marcu, 2003). In our work, we focus on both starting by retrieving a fixed number of passages that are most likely to answer the user's question, and then given the retrieved passages we try to re-rank them in order to return the top ranked passage from a ranked list of relevant passages. In this section, we review the

most important approaches in the literature related to passage retrieval and ranking for answer selection.

3.2.1 Lexical matching

A quantitative evaluation was conducted by Tellex et al. (2003) of different PR algorithms applied to QA based on lexical matching. The evaluated systems are the following:

- The BM25 system (Robertson et al., 1996) employed a PR algorithm based on a sliding window on the document to detect the passages. Then, a score is assigned to each passage based on the proposed Okapi BM25 similarity between the question and the passages.
- The MultiText system (Clarke et al., 2000) employed a density-based PR algorithm that favors short passages containing terms with high idf values. In this algorithm, each passage is a window that begins and ends with a question term, and its score is based on both the number of query terms in the passage and the window size.
- IBM system (Ittycheriah et al., 2000) computed a linear combination score based on the following distance measures for the passage: the *matching words measure* which sums the idf values of passage words that overlap with the question, the *thesaurus match measure* which calculates the idf values of query words whose WordNet [1] (Fellbaum, 1998) synonyms occur in the passage, the *mismatch words measure* which sums the idf values of words that appear only in the query, the *dispersion measure* which considers the number of words in the passage between matching query terms, and the *cluster words measure* which counts the number of words that appear adjacently in the question and as well as in the passage.
- The Mitre system (Light et al., 2001) used the top 10 documents returned by the AT&T search engine and ranks each passage according to the number of words that occur in both the passage and the query.
- SiteQ system (G. G. Lee et al., 2001) used a PR algorithm which calculates the score of a passage by summing the weights of its individual sentences. The weight of a sentence is based on query term density.
- ISI system (Hovy et al., 2001) employed an algorithm which ranks passages according to their similarity to the query by weighting different features: match of proper names, match of question terms and match of stemmed words.

[1] WordNet is a large-scale lexical database of english with over 70,000 entries, structured as a semantic network with terms linked by the traditional ISA connections including many types of links such as hypernym, antonym, holonym, and meronym connections.

- IR-n Alicante system (Llopis & Vicedo, 2002) used a PR algorithm which calculates the normalized cosine similarity between query words and the passage taking into account the number of occurrences of a term in both the passage and the query as well as their idf values.

This quantitative evaluation revealed that the choice of the applied document retrieval engine affects the performance of PR algorithms. The best evaluated algorithms used density-based measures for scoring question terms and boolean querying schemes perform well in the context of QA task. However, all the mentioned algorithms process each term of the question as an independent symbol and take into account neither the order of words nor their dependency relations. Therefore, many irrelevant passages share the same query terms with those which are correct but the dependency relations between these terms may be different from those existing in the question. In order to address this limitation, there have been several attempts to consider term dependencies such as (Srikanth & Srihari, 2002). In most cases, bigrams were used to replace unigrams with the simple assumption that two adjacent terms are related. Nevertheless, dependencies not only occur between adjacent words but may also exist between distant words. Within this context, Gao et al. (2004) proposed a general dependence language model to relax adjacency constraint, where a hidden variable called "linkage" was integrated to model the term dependencies within the query as an acyclic planar undirected graph.

3.2.2 Syntactic matching

Some studies relied on the syntactic matching, particularly the syntactic dependencies instead of simple keywords. Therefore, two main approaches were used, namely strict matching and fuzzy relation matching.

The former one searches for an exact matching between the dependency relationships of the question and those of a passage. For example, authors in (Katz & Lin, 2003) resorted to this approach using ternary expressions (e.g., <subject, relation, object>) to deal with two critical phenomena that cannot be simply handled by linguistically QAS: semantic symmetry and ambiguous modification. A relation is semantically symmetry if there exist three words $w1$, $w2$ and $w3$ where $S(R(w, w1)) = S(R(w2, w)) = 1$. In other words, we can swap the subject and the object and still have correct relations. For example, the answers to the questions "What do frogs eats?" and "What eats frogs?" are likely to contain the keywords "frog" and "eat". As "eat" is a semantically symmetry relation, both (eat (frog, x)) and (eat(x, frog)) may be found in the corpus which cause confusion. Many other verbs could cause the same problem such as *beat, visit, meet, etc*. Otherwise, a word w, involving a relation R, is an ambiguous modifier if there exist at least two words $w1$ and $w2$ in the same local context as w, where the two relations $R(w, w1)$

and $R(w, w2)$ are correct: $S(R(w, w1)) = S(R(w, w2))$=1. For instance, for the sentence "The planet's largest volcanoes", the adjective "largest" causes ambiguity between (largest, planet) and (largest, volcanoes). That is to say, given passages with similar lexical content containing the adjective "largest", we cannot easily know which head noun this adjective is modifying.

The second approach relied on a fuzzy relation matching, searching for a non exact matching between the dependency relationships of the question and those of a passage. In this context, Cui et al. (2005) used a dependency parser to extract typed relations and generate dependency trees that relate to the question and the passages where nodes represent words and edges denote the relations labels. Then, the paths between nodes and edges are extracted and their matching scores are measured. The matching score of a candidate sentence is considered as the probability of translating its relation path to that of the question. Nonetheless, this process requires a relation mapping model which can be acquired by two statistical methods: one based on mutual information (MI) and the other based on expectation maximization (EM). Added to that, this approach requires a training set for learning which is not always available.

The major shortcoming of syntactic matching is the need of a syntactic parser which is not always available in every language. It also requires adaptation and the performance of QAS significantly depends on the performance of the analyzer.

3.2.3 Semantic matching

Several works based on semantic matching have been proposed in the past using semantic resources to generate semantic relations or semantic annotations for the query and the passages. For instance, Tari et al. (2007) used various entity recognizers along with semantic relatedness based on MeSH [2] ontology and UMLS [3] semantic network to verify the relevance of candidate passages. Recognition of entities was applied for the candidate passages using a statistical learner to identify keywords with entity classes of interest. Unlike semantic similarity which is the relatedness of a pair of terms that belong to the same class, (e.g., a is-a relation), semantic relatedness refers to terms that are related but do not necessarily belong to the same class. This approach is based on the hypothesis that using only the general keywords as well as their expanded forms in the queries could significantly affect the precision of retrieval since many terms can be related to one general keyword but they are not all relevant to the input question.

Ofoghi and Yearwood (2009) proposed another semantic matching based approach, where

[2]Medical Subject Headings thesaurus is a controlled vocabulary produced by the U.S. the National Library of Medicine.

[3]Unified Medical Language System is a thesaurus of biomedical concepts designed and maintained by the US National Library of Medicine. It consists of knowledge sources and a set of software tools and it also provides facilities for NLP.

they assessed the impact of using semantic class overlap evidence in the PR effectiveness of QAS. They employed FrameNet [4] frames to capture the semantic class of each term in the question as well as in the passages. To this purpose, two methods were tested: The first one combines passage scores got from a baseline PR system with those obtained through correspondence of semantic classes. The second one integrates the correspondence of semantic classes in computing the score of the retrieved passages.

Bilotti et al. (2010) have carried out a general rank-learning framework for passage ranking for QA using linguistic and semantic features. The framework analyzes the question and checks of complex and deep linguistic and semantic constraints over keywords and represents them as an annotation graph. These constraints involve named entity features and features derived from semantic role labeling. Nevertheless, the inconvenience of this method is that it relies on manual design of features.

Within the same context, Araki and Callan (2014) proposed an annotation similarity model to improve passage ranking for QA. Historical fact validation is a subtask introduced by authors to indicate whether a candidate sentence gives historically correct information according to a reference to Wikipedia as a reliable information sources. A three-stage passage ranking model is built for historical fact validation. The former stage is document retrieval using query comprising all raw words in the sentence to guarantee high recall. The second stage is PR which doesn't involve any type of annotation, where each retrieved document is segmented into a restricted number of passages by a fixed-length window. Note that tf-idf was used for retrieving in both stages. The third stage is passage ranking where a list of annotation is involved and two similarity models were combined: a bag-of-words similarity model and an annotation similarity model. To the best of our knowledge, this is the first work using linguistic and knowledge-based resources in passage ranking in an unsupervised manner. Notwithstanding the fact that it has shown promise in terms of precision and MRR, it suffers from a sparseness problem of semantic arguments as only a few number of semantic role annotations is produced by the system. The major shortcomings of this approach is the difficulty of mapping and matching predicates and arguments to model relations and the dependence of the result on the relevance of the retrieved passages, thus it would be better to moved up the approach into the initial process of selecting passages candidates.

Although these semantic matching based approaches allow to detect genuine passages, they require semantic resources (e.g., FrameNet) which mostly cover neither all domains nor all terms.

[4] http://framenet.icsi.berkeley.edu/

3.2.4 Syntactic and semantic matching

Further attempts have been made in combining both semantic and syntactic matching in the context of PR to take advantage of both techniques. In this context, Laurent et al. (2005) developed the well known commercial system QRISTAL [5] based on syntactic and semantic analysis of the text to perform multi criteria indexing and answer extraction. Based on information get from the analysis of passages, numerous indexes are made such as domain index, keyword index, index concept, etc. Then, when the user asks his question, this latter is analyzed in a deeper way than that established for the passages because the question is usually short and ambiguous. Once this analysis is done, the indexes are consulted as well as the top ranked blocks relative to these indexes are reanalyzed, but this time with computing a weight for each sentence for the classification of answers. This weight is based on the number of words and named entities in this sentence, the presence or absence of the response type corresponding to the question and the agreement between the themes and the domains.

In addition, InSicht is a QAS implemented for German based on a deep syntactico-semantic analysis of questions and documents (Hartrumpf, 2005). Documents are first analyzed using a syntactico-semantic parser WOCADI [6] and each sentence is represented by a semantic network. Likewise, the question is analyzed by the same parser and represented by a semantic network pointing out some additional information such as the question type, the focus, etc. Subsequently, query expansion is applied to generate equivalent semantic networks for the purpose of finding implicit answers, utilizing among others, paraphrase rules and lexico-semantic relations (e.g., synonymy, hyponymy). Semantic networks are further simplified and normalized. Finally, the simplified semantic network of the question as well as the question type are used for the matching task. Therefore, the system returns German passages as candidate answers which will be then the input of the answer selection module.

Within the same context, Shen and Lapata (2007) utilized syntactic information under dependency relation paths and FrameNet semantic roles. More precisely, the predicates relative to the question as well as the passages (e.g., verbs, nouns, adjectives) are defined and for each frame, the related semantic roles are produced which can be then represented as a complete bipartite graph where each frame element is connected to the possible semantic roles allowed by the predicate. Then, the similarity between a question and its candidate answer is calculated by matching their predicates and semantic role assignments which are represented by graphs. Thus, the task can be modeled as a graph matching problem.

[5] A QAS based on NLP developed with the support of ANVAR (Agence nationale de valorisation de la recherche) and the European Commission (TRUST , IST-1999-56416), cf.

[6] WOrd ClAss based Disambiguating is a syntactico-semantic parser that transforms articles into semantic networks

Severyn et al. (2013a) applied a learning to rank model to learn complex patterns, for instance, learning the relational semantic structures which appear in questions as well as their passages. To this end, the learning to rank algorithm was fed a tree representation derived from a representation of the question and answer passage pair following the approach in (Severyn & Moschitti, 2012) using both shallow syntactic trees and relational nodes (i.e., those matching the same words in the question and in the passages). Then, a large baseline for passage re-ranking was established with a larget sample of different features. To establish relational links (Severyn et al., 2013b) between a question and a passage, focus classifiers based on kernel methods were implemented to be merged with a named entity recognizer (NER). The focus classifier identified the term of the query to be linked to the named entities of the passage according to the compatibility of their categories. For example, a named entity of type PET is compatible with a category of a question asking for an ANIMAL. Even though this method is efficient, it relies on a large set of semantic topic models such as WordNet and word alignment which require a deep study on how to exploit them at their best.

Besides, in (Moschitti & Quarteroni, 2011), supervised discriminative models such as Support Vector Machine (SVM) which learn to rank sentences taking advantage from examples of question and answer pairs, were studied by applying structural kernel functions as powerful generalization methods to exploit syntactic and semantic structures. SVM is a supervised learning model that was largely applied in the context of ranking passages and it will be overviewed in the next Chapter. Sequence kernels were modeled for words and part-of-speech tags which capture lexical semantics and syntactic information. Then, tree kernels were applied to encode deeper syntactic and semantic information. Although this approach has shown significant feasibility which is due to the SVM method which makes the learning algorithm robust even to numerous irrelevant features, it remains computationally expensive due to the application of SVM and kernel methods.

3.2.5 Contextual matching

Otherwise, some works used the context of words as a simple intuition for ranking passages. For instance, Toba et al. (2010a) proposed a contextual approach for passage selection which aims to find the supporting word context from question and candidate passages during the passage selection using the techniques of state-of-the art QASs i.e., Open Ephyra [7], JIRS [8], and the Semantic Vectors [9] open source packages. This latter is an open source package that can be employed to create context vectors of word concepts in a specific domain and apply random

[7] http://www.ephyra.info

[8] http://sourceforge.net/projects/jirs/

[9] http://semanticvectors.googlecode.com/

projections of words in the collection of documents to implement word space methodology. In semantic vectors, words are represented as vectors where those having related meanings are in close proximity. Then, the random projection is called to build the term vectors as well as the documents vectors (Widdows & Ferraro, 2008). This approach has improved the empirical performance of the passage selection but only in a specific domain due to the nature of the semantic vectors applied.

Within the same context, Yen et al. (2013) introduced a machine learning-based QA framework which integrates context-ranking model named (CRM) that re-rank the passages retrieved from the initial retrieval engine to find the appropriate answers. The given model uses contextual features of proper names combined in an SVM model to determine whether a candidate passage is relevant to the query type. More precisely, each named-entity word in the passage is detected and centralized to extract the corresponding context features in a fixed-size window, namely the form of the word, its part-of-speech tag, its named entity class, its match degree with the question terms and its token class. However, in the proposed framework, the performance of the model is highly dependent on the question classification. We emphasize that our passage re-ranking model is broadly inspired from this work to the extent that we use SVM to incorporate different features extracted from the passages, nonetheless, unlike Yen et al. (2013), we are constrained neither by Named Entity words nor by a fixed-size window. We are going beyond a simple detection of the NE class, the form and the part-of-speech tag of the word to calculate lexical, semantic and n-gram similarity measures.

3.2.6 N-grams

Other works were going beyond the simple lexical matching, resorting on a different model for the purpose of PR called n-grams (Majumder et al., 2002), which refers to sequences of consecutive items (characters or words) extracted from a given text. N-grams is a powerful and fast tool that does not deal with terms as independent symbols but it takes into account the simple dependency between them. In this context, Radev et al. (2005) developed a probabilistic method for Web-based Natural Language QA. The process is as follows: the web documents are first retrieved by a search engine are segmented into passages. Then, these latter are ranked using a score based on n-grams calculated as follows:

$$Score = \frac{w_1 \sum_{i=1}^{N_1}(tf_i \times idf_j) + w_2 \sum_{j=1}^{N_2} tf_j + w_3 \sum_{K=1}^{N_3} tf_k}{normalized\_Factor} \tag{3.1}$$

where $N_i(i = 1, 2, 3)$ is the total number of occurrences of unigram, bigram, and trigram in a passage, $w_i(i = 1, 2, 3)$ is the linear combination of weights assigned according to the importance of n-grams, tf_i denotes the term frequency of the i-gram. idf measures the rarity of a unigram. Normalized_Factor is a normalization factor that depends on the passage size.

In addition, Correa et al. (2010b) proposed an n-gram based PR system designed for QA using JIRS [10] (JAVA Information Retrieval System) described in (Gómez et al., 2007). This system first indexes and segments the set of documents into snippets and then extracts candidates passages using keywords. Subsequently, it introduces an n-gram based model to select and rank the most relevant passages. Once had the n-grams of both the question and the passages, a similarity measure is applied to compare them favoring the passages containing more query n-grams and longer ones. In their work, the similarity between a passage and the question is given by:

$$Sim(p,q) = \frac{\sum_{j=1}^{n} \sum_{x \in Q} h(x, P_j)}{\sum_{j=1}^{n} \sum_{x \in Q} h(x, Q_j)} \tag{3.2}$$

where Q_j denotes the set of j-grams of the question, P_j denotes the set of j-grams of the passage, $j = [1..n]$ with n is the number of terms of the question and $h(x, P_j)$ represent the weight for the j-gram x. This weight is set to:

$$h(x, P_j) = \begin{cases} \sum_{k=1}^{|x|} W_x & \text{if} x \in Q_j \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

where w_k is the weight of a term k of the n-gram x depending on the number of passages n_k in which the term occurs and N denotes the total number of passages. This weight is determined as follows:

$$w_k = 1 - \frac{log(n_k)}{1 + logN} \tag{3.4}$$

The same approach was followed by Buscaldi et al. (2010) with a difference in the applied n-gram model. This latter is used to measure the similarity between the question and the passages considering the n-grams of the passages existing in the question and their proximity. The passages having more n-grams in common with the question and whose n-grams are closer are favored. The weight of an n-gram is divided by the distance between this latter and the n-gram of maximum weight.

3.3 Summary

A QAS can be viewed as an embedded passage retrieval process bookended by IR and NLP techniques that allow to understand the human question, on the front end, and post-retrieval, to

[10] http://sourceforge.net/projects/jirs/

locate relevant answers. Owing to its importance, PR has been widely studied by researchers over recent years using various different approaches.

In this Chapter, we have reviewed the major state-of-the-art PR approaches ordered by the matching type adopted: lexical, syntactic, semantic, contextual matching and n-grams. Regarding the former matching type, a quantitative evaluation of different PR algorithms applied to QA realized by Tellex et al. (2003) has shown that the DR module affects the performance of PR algorithms as PR and DR are dependent to each other.

Additionally, we have deduced that the best algorithms studied used density-based measures for scoring question terms and also, boolean querying schemes performed well in the context of QA task. However, all the mentioned algorithms process each term of the question as an independent symbol and do not consider the order of words and their dependency relations. Therefore, many irrelevant passages share the same query terms with those which are correct but the dependency relations between these terms are different from those existing in the query. In order to overcome this shortcoming there have been several attempts to consider term dependencies such as (Srikanth & Srihari, 2002). In most cases, bigrams are used to replace unigrams with the basic assumption that two adjacent terms are related, but dependencies not only occur between adjacent words but may also exist between distant words. Furthermore, numerous works has relied on syntactic matching, in particular syntactic dependencies instead of simple keywords. As a matter of fact, two main approaches were used: strict matching and fuzzy relation matching. The major shortcoming of these approaches is the need of a syntactic parser which is not always available for any languages. Besides, it requires adaptation and the performance of QAS depends on the performance of the analyzer.

Moreover, works based on semantic matching relied on semantic resources to generate semantic relations or semantic annotations for the query and the passages. Notwithstanding these approaches allow to detect genuine answers, they require semantic resources which mostly cover neither all domains nor all terms. In order to exploit the benefits of both approaches, there have been an abundance of works combining semantic and syntactic matching in PR. Other works resorted to n-gram technique to retrieve passages which goes beyond the above mentioned simple lexical matching. The strength of n-grams which refer to sequences of consecutive items is that it does not deal with terms as independent symbols but it takes into account the simple dependency between them. Recall that we will resort to this powerful technique in our PR model. Besides, most context-based approaches lack flexibility as they often did not allow words to belong to more than a topic and they are more adequate to some questions than others. Added to that, such approaches are often highly dependent on the question classification module and cover only a specific domain due to the nature of the semantic vectors applied.

On the other hand, most context-based approaches lack flexibility as they often did not allow words to belong to more than a topic and they are more adequate to some questions than

others. Added to that, such approaches are often highly dependent on the question classification and cover only a specific domain due to the nature of the semantic vectors applied. Regarding pattern-based approaches, most of them require a lot of training data and the identification of the candidate answers is a-high precision- task which is always easy neither for all cases, nor for all languages. Otherwise, SVM was successfully applied in ranking passages, but the major inconvenience of this technique is that it is essentially a binary classifier, which requires dividing multiclass into numerous binary categories. Most existing multiclass SVM models (P.-C. Wu et al., 2008) could not yet offer an efficient time performance. Moreover, SVM need a large training set which contains sufficient examples for answer selection and also the choice of the features used in such classifiers is a critical task.

3.4 Related work on question retrieval in cQA

The problem of QR has been intensively approached in recent years in order to improve user satisfaction by reducing the time lag required to get an answer. It is assumed that the associated answers to the similar questions can respond to the new query. Traditional QR research mostly focuses on factoîd questions which differ from open questions in that they are direct and rarely include noise. However, the open questions posted on community forums are generally not grammatically correct, neither formal, and result in noisy texts. In community forums, similar questions can widely vary in length, vocabulary, style, and content quality, which makes the problem of QR a challenging one mainly due to the word mismatch issue.

3.4.1 Basic Models

Among the basic models, we briefly present the most used ones for the question retrieval task namely, Vector Space Model, Okapi BM25Model, Query Likelihood Language Model and Translation Model.

Vector Space Model:

The Vector Space Model referred to as VSM has been extensively used for the VSM task to calculate the cosine similarity between questions (Duan et al., 2008; X. Cao et al., 2010). A widely used variation of VSM model sets the similarity score of query q and question d as follows:

$$Sim_{VSM}(q,d) = \frac{\sum_{t \in q \cap d} w_{q,t} w_{d,t}}{W_q W_d} \tag{3.5}$$

where $w_{q,t}$ and $w_{d,t}$ determine respectively the IDF of a term t in the collection and captures the TF of a term t in the question d. They are calculated as follows:

$$w_{q,t} = ln(1 + \frac{N}{df_t}) \tag{3.6}$$

$$w_{d,t} = 1 + ln(tf_{t,d}) \tag{3.7}$$

while W_q and W_d are calculated as follows:

$$W_q = \sqrt{\sum_t w_{q,t}^2} \tag{3.8}$$

$$W_d = \sqrt{\sum_t w_{d,t}^2} \tag{3.9}$$

The main limitation of VSM is that it favors short questions. As shown in equation 3.5, short questions having small W_d will obtain higher similarity scores when other conditions are equal. However, cQA services can handle a wide range of questions not limited to factoîd ones.

Okapi BM25Model:

In order to address the shortcoming of VSM, Okapi BM25 (BM stands for Best Matching) model takes into consideration the question length (Jeon et al., 2005). Okapi BM25 is the most widely applied model among a family of Okapi retrieval models proposed by (Robertson & Walker, 1994) which has proven significant performance in several information retrieval tasks. Basically, Okapi BM25 is one of the most robust and effective retrieval functions used to rank questions according to their relevance to a given query. One well-known version of Okapi BM25 model (X. Cao et al., 2010) is given in equation 3.10 below:

$$Sim_{BM25}(q, d) = \sum_{t \in q \cap d} w_{q,t} w_{d,t} \tag{3.10}$$

where

$$w_{q,t} = ln(\frac{N - df_{t+0.5}}{df_t + 0.5}) \frac{(k_3 + 1)tf_{t,q}}{k_3 + tf_{t,q}} \tag{3.11}$$

$$w_{d,t} = \frac{(k_1 + 1)tf_{t,d}}{K_d + tf_{t,d}} \tag{3.12}$$

$$K_d = k_1(1 - b + b\frac{W_d}{W_A}) \tag{3.13}$$

In these latter equations, k_1, b, and k_3 are parameters. W_d denotes the question length or the number of words of the question d and W_A represents the average question length in the entire collection. Note that in this model, the parameter b determines the influence of question length on similarity score.

Query Likelihood Language Model:

The Query Likelihood Language Model (QLLM) is a language model widely used in information retrieval (Zhai & Lafferty, 2004) where the key idea is to construct for each question a language model, and then rank the questions according to the probability $P(d \mid q)$ which is estimated as the likelihood that a question is relevant to the query. Using Bayes' rule, the probability of a question, given a search query is calculated as follows:

$$P(d \mid q) = \frac{P(q \mid d)P(d)}{P(q)} \tag{3.14}$$

It is worthwhile to note that the probability of the search query $P(q)$ can be ignored since it is the same for all questions. The prior probability of a candidate question P(d) is often considered as uniform, and therefore, it can also be ignored. According to the Language Modeling approach, the archived questions are ranked by the probability that a search query would be considered as a random sample from the respective question model. The multinomial unigram language model is the most commonly used way to realize this. We have:

$$P(q \mid M_d) = K_q \Pi_{t \in V} P(t \mid M_d)^{tf_{t,d}} \tag{3.15}$$

where $K_q = L_d!/(tf_{t_1,d}!tf_{t_2,d}!...tf_{t_M,d}!)$ is the multinomial coefficient for the query q, and $L_q = \sum_{1 \leq i \leq N} tf_{t_i,q}$ is the query length given the term frequencies tf in the query vocabulary N. Note that the multinomial coefficient which is often removed from the calculation since it is a constant for a particular query. However, QLLM model might not be effective when there are few vocabulary between q and d. For example, *"What are the home remedies for yellow teeth?"* and *"I want to whiten my teeth naturally, what should I do?"* are two semantically similar questions but only have few vocabulary overlaps which lead to a poor similarity estimation for QLLM.

Translation Model:

The translation model referred to as TM takes advantage of word-to-word translation probabilities in the language modeling framework to overcome the vocabulary mismatch problem faced by QLLM. Existing work (Duan et al., 2008; Xue et al., 2008) demonstrates that TM obtains significant performance for QR. Following the TM based approach presented by (Jeon et al., 2005), the ranking score $Sim_{TM}(q, d)$ of a query q and a question d is calculated as follows:

$$Sim_{TM}(q, d) = \prod_{t \in q} (1 - \lambda) \sum_{w \in d} T(t \mid w) P_{ml}(w \mid d) + \lambda P_{ml}(t \mid Coll) \tag{3.16}$$

where $T(t \mid w)$ represents the probability that a word w is the translation of word t. In the retrieval task, the latent topic information was often ignored when computing the semantic similarity between questions. In order to enhance the performance of the TM model for QR, (G. Zhou et

al., 2011) proposed a topic model that captures the latent topics in the content of questions in modeling the translation of phrases. The semantic similarity based latent topics is then combined with the translation-based language model into a unified framework. Another line of work on translation models focused on extending the word-based translation model to incorporate semantic concepts and exploring different strategies to learn the translation probabilities between words and concepts using the cQA archives (Singh, 2012). However, the main limitation of this approach is the need of huge training corpora which is so costly to create. Besides, machine translation model usually works less well for language pairs with great grammatical differences and very different word order.

3.4.2 Category information:

Certain cQA services such as Yahoo! Answers often organize questions into a hierarchy of categories which offer useful extra knowledge for question retrieval. For instance, the subcategory "Environment.conservation" is a child category of "Environment" in Yahoo! Answers hierarchy of categories. A user, when posting a question is required to pick out a category label for its query from a predefined list of categories. The questions belonging to the same category or subcategory are often related to the same general subject. In order to simplify the basic idea behind category information in QR, we give the following query q: "could you recommend best destinations for honey moon in Europe ?", where the user is interested in destinations particularly in Europe. Therefore, the question d "could you recommend best destinations for honey moon in Asia?" is not relevant to the first question q, despite the fact that the two questions are syntactically close. Thus, the connection between the question q and the category "Travel.Europe," might promote the questions ranking in that category and then enhance the QR performance.

To the best of our knowledge, there are only two attempts that have exploited the available category information for question retrieval. In the first one, (X. Cao et al., 2009) consider two different approaches to using categories for improving the performance of language-model based QR. The former approach resorted to classifiers to calculate the probability of a query belonging to different categories. Then, this probability is employed to set the ranking returned by the language model. Nevertheless, the experiments showed that this approach led to a slight improvement in language model. The second approach relied on a two-level smoothing model, where a category language model is calculated and then smoothed with the entire question collection. Hence, the question language model is smoothed with the category model. Although this second approach has proven to significantly improve the performance of the LM for QR, the use of category information was restricted to the LM. In order to overcome this limitation, authors in (X. Cao et al., 2010), proposed a general approach named category enhanced retrieval model to exploit category information that can be applied to any QR model. The basic idea is that basi-

cally, the more a category is related to a given query q, the more likely that this category includes questions similar to q. The given approach ranks a historical question d based on a combination of two relevance similarity scores: a global relevance score denoted as $S_{q,cat(d)}$ which represents the similarity between query q the d 's category (cat(d)), and a local relevance score denoted as $S_{q,d}$ between q and d within d's category. The final similarity score is then calculated after normalizing both the global relevance and the local relevance scores as follows:

$$Sim_{CI}(q,d) = (1-\beta)N(S_{q,d)} + \beta N(S_{q,cat(d)}) \tag{3.17}$$

where β is the weighting parameter and $N(S_{q,d})$ is the normalization function.

3.4.3 Syntactic Tree Matching

The major reason that makes the QR task notoriously complex, is that the questions posted by community users in human natural language have different lexical, syntactic and semantic features. For instance, the questions "How can I learn Chinese in a short period?" and "Are there any fast methods of learning Chinese in a few weeks?" are semantically similar asking for the same matter but it is difficult for an automatic system to detect the similarity between these two questions since they neither share many common words nor have the same syntactic structure, which make it difficult for an automatic system to detect the similarity between these two questions. Accordingly, the basic techniques based the bag of-word (BoW) approach such as VSM and QLLM may become ineffective and work poorly. Therefore, in these circumstances, more robust methods based on syntactic or semantic features become essential and strongly required. Within this context, (Wang et al., 2009) introduced a syntactic tree matching (STM) approach to find similar questions using syntactic features. In nutshell, given the parsing trees T_1 of search a query q and T_2 of a question d, the similarity score between T_1 and T_2 trees is computed as follows:

$$Sim_{STM}(q,d) = \frac{S(T_1,T_2)}{\sqrt{S(T_1,T_1)S(T_2,T_2)}} \tag{3.18}$$

$$S(T_1,T_2) = \sum_{r_1 \in T_1} \sum_{r_2 \in T_2} M(r_1,r_2) \tag{3.19}$$

where $M_{(r_1,r_2)}$ denotes the matching score of two nodes r_1 and r_2 calculated as follows:

$$M_{(r_1,r_2)} = \begin{cases} \delta_{r_1}\delta_{r_2}\lambda^{S_{r_1}+S_{r_2}}\mu^{D_{r_1}+D_{r_2}} & \text{if } r_1 \text{ and } r_2 \text{ are terminals} \\ \delta_{r_1}^{\eta}\delta_{r_2}^{\eta}\lambda^{2\eta}\mu^{n[2-(1+nc(r_1))(D_{r_1}+D_{r_2})]} \times F & \text{otherwise} \end{cases} \tag{3.20}$$

where in this equation, δ_{r_1} represents the importance of node r_1 in the tree T_1, while S_{r_1} and D_{r_1} denote respectively the size and the depth of the tree fragment with the root node r_1. Note

that S_{r_1} is defined by the number of nodes that the tree fragment includes and D_{r_1} represents the level of the tree fragment root in the global syntactic parsing tree. $nc(r_1)$ means the total number of children of the node r_1. λ is a parameter representing the preference between size and depth while μ is another parameter denoting the total number of matched tree fragments. The function F in the above equation is computed as follows:

$$F = \prod_{j=1}^{nc(r_1)} M(ch(n_1, j), ch(n_2, j)) \tag{3.21}$$

where $ch(n, j)$ in this function denotes the j^{th} child of node n in the tree.

3.4.4 Latent Semantic Indexing

Latent Semantic Indexing (LSI), also known as Latent Semantic Analysis (LSA) (Deerwester et al., 1990), is a widely used approach in natural language processing, that supposes that there is somehow a latent structure in term usage that is usually hidden due to the variability in word choice. The basic idea behind the LSI model is to analyze relationships between a collection of documents and the terms they contain by mapping them to a latent semantic space made by the set of concepts related to the documents and terms. The retrieval task can be performed using vectors obtained from the truncated SVD which is utilized to estimate the word usage structure through documents. This approach attempts to overcome the lexical gap problem in information retrieval by employing statistically derived conceptual indices rather than individual terms. (Qiu et al., 2013) relied on LSI to improve QR in cQA by proposing a model based on LSI with tensor analysis called LSTI, which can detect word associations among various parts of cQA triples simultaneously. The core idea of the proposed model is to not consider the question as a whole like in the previous cited models, but rather consider the real components of a question. Accordingly, the question can be represented with a triple form $\prec$ question title, question content, answer content $\succ$. More concretely, for a collection of cQA triples, $\prec q_i, c_i, a_i \succ (i = 1 \ldots k)$, where q_i is the question and c_i and a_i denote respectively the content and answer of the question and K is the number of singular values of entries, a 3-order tensor $D \in \mathbb{R}^{K \times 3 \times T}$ is used to represent the collection of T terms. Given a search query q and a candidate question d, their corresponding triples respectively $D_q \in \mathbb{R}^{1 \times 3 \times T}$ and $D_d \in \mathbb{R}^{1 \times 3 \times T}$ will be projected to term space. The similarity score is then calculated as the normalized Forensics inner product of their projection matrices. Although LSTI has proven effective in helping overcome the semantic gap problem in QR, this model faces the problem of data sparsity in term space since the size of a tensor in LSTI is larger than a term document matrix in LSI.

3.4.5 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is a popular generative probabilistic model that describes how text data are generated, in terms of topics. The basic idea is that every document is about several topics and that each word in the document can be associated with one of these topics. Topic models allow to compute how much the document is about the considered topics by calculating topic proportions for every document and producing list of words as topic representation.

Within this context, (Cai et al., 2011) proposed a topic model integrated with the category information into the process of detecting the latent topics in the content of questions and calculate the semantic similarity between questions based on the latent topic information. To solve the word mismatch problem, a translation-based language model was used to extract knowledge from question answer pairs which are collected from a cQA platform. A latent topic model was employed to derive knowledge from the distribution of words and categories in cQA archives assuming that the two knowledge are complementary.

Ji et al. (2012) have proposed an approach to question retrieval based on the question-answer topic model which uses LDA to learn the latent topics underlying the text of question-answer pairs. Nonetheless, their topic model is designed at word level, and thereby the topics in the model are not general and expressive enough.

Chen et al. (2016) reported a more general approach to alleviate the lexical gap problem in QR. They present a hybrid approach that blends certain language modelling techniques, namely the basic query-likelihood language model, the translation-based language model, and their proposed semantics-based language model. The semantics of each question was presented by a probabilistic topic model which employs local and global semantic graphs for detecting the hidden interactions among entities (e.g., places, people and concepts) in question-answer pairs.

By and large, the major shortcoming of topic modeling schemes, such as LDA and LSA, is that they need to make modifications when applied on short texts through aggregation strategies. Indeed, short texts carry limited context information, which leads to sparsity problems when applying conventional topic models.

3.4.6 Neural Networks

Recent works on QA focused on the representation learning for questions, relying on an emerging model for learning word representations in a low-dimensional vector space called Word Embeddings. Word embedding models turn words into vectors, where the similarity between the word vectors reveals the semantic and syntactic similarities between the corresponding words.

Lately, there has been burgeoning interest in word embeddings which have been shown to achieve impressive performance in several NLP tasks (Turian et al., 2010; Collobert et al., 2011), in particular for question retrieval (G. Zhou et al., 2015). The main advantage of these emergent unsupervised learning models is that they do not require expensive annotation, but can be derived from a large-scale unannotated collection. They can then be applied in tasks where only small amounts of labeled data are available.

Distributed representation of words as continuous vectors has been recently addressed by Mikolov, Sutskever, et al. (2013) who presented powerful neural network models to learn word representations, including two unsupervised models learned from huge corpora namely, the skip-gram model and the continuous bag-of word model. There has been further a host of work examining the task of learning word representations such as (Maas et al., 2011; Huang et al., 2012). However, major previous works are mainly based on the word co-occurrence information and therefore the yielded word vectors cannot detect the relationship between two semantically equivalent terms if they have little context information. In order to improve learned word embeddings, Yu and Dredze (2014) proposed the use of a prior knowledge contained in semantic resources, while C. Xu et al. (2014) proposed to incorporate knowledge graphs into the learning process assuming that it gathers useful relational knowledge that encodes the relationship between entities and categorical knowledge that encodes the properties of entities. Otherwise, G. Zhou et al. (2015) was the first work to learn continuous word embeddings with metadata of category information within cQA corpora for the QR task.

Since we believe that the word representation is crucial for the QR task and inspired by the success of the latter model, we rely on word embeddings to detect semantically similar questions in cQA. Along with the popularization of word embeddings owing to its capacity to produce distributed representations of words, various neural network architectures such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) have proven to be efficient in extracting higher-level features from constituting word embeddings. For instance, Dos Santos et al. (2015) used CNN along with bag-of-words (BOW) representations of the questions to compute the cosine similarity scores. Within the same context, Mohtarami et al. (2016) proposed a bag-of-vectors approach using CNN and attention-based LSTM to capture the context and semantic similarity between the community questions and rank them accordingly. LSTM model was also employed in (Romeo et al., 2016) with an attention mechanism to detect long dependencies in community questions. Interestingly, the weights learned by the attention model were utilized for picking up significant segments and thus enhancing syntactic tree-kernel models. More recently, Kamineni et al. (2018) turned the QR task into a binary classification problem using a combination of LSTM and a contrastive loss function to effectively memorize the long term dependencies. In our work, we resorted to a Siamese adaptation of LSTM (Mueller & Thyagarajan, 2016) for pairs of variable-length sentences, which has accomplished excellent outcomes in the semantic text similarity task and inspired us in our QR

problem.

Despite being effective and suitable for complex NLP problems, NNs has some downfalls to go along with its benefits. The most known shortcoming of NNs is debuggability due to their black box nature, that is to say we do not know how and why our NN came up with a certain output. Thus, if something fails, it is hard to debug and fix the problem. Indeed, when you have features that are human interpretable like Decision trees, it is easier to understand the source of a problem. Moreover, NNs mostly require large datasets for training which can make computational requirement sometimes prohibitive.

3.4.7 Summary

Finding semantically similar questions can not only help to exploit the information available in the tremendous community archives but also be an intermediate step in QA. QR aims to reduce the lag time incurred by waiting for new answers, thus improving user satisfaction. Over time, the task of QR has been approached by numerous researches in different ways. The ultimate goal of the research was to address the lexical gap problem between the questions in cQA.

In this section, we reviewed the main proposed approaches to address the QR task grouped by the employed model. Early works relied on the basic models, such as Vector Space Model, Okapi BM25Model, Query Likelihood Language Model and Translation based Model. However, such models mostly do not take into account the question length and usually do not operate well for language pairs with great grammatical differences and very different word order. Other works made use of the category information assuming that the questions belonging to the same category or subcategory are often related to the same general subject but, the use of category information was restricted to the language model. In addition, some attempts have been made relying on syntactic features to find similar questions using syntactic features. Nevertheless, such an approach requires a parser, which is not always available for any language, and its outcome significantly depends of that of the parser which mostly cannot derive deep and complex syntactic structures. There has also been a host of works investigating the use of topic modeling in the QR task mainly LSI and LDA. Although topic models have proven effective to overcome the semantic gap problem, the major limitation of such models is that they need significant modifications when applied on short texts through aggregation strategies. Moreover, short texts carry limited context information, which leads to sparsity problems when applying conventional topic models. Otherwise, more recent works relied on NNs which become be all and end all of most NLP tasks. Notwithstanding their outstanding outcomes, NNs require a huge amount of training data, which can make computational requirement sometimes prohibitive, and suffer from two major issues, namely interpretability and debuggability due to their black box nature.

It is worth mentioning that work on cQA has mostly been carried out for other languages than Arabic due to a lack of available Arabic resources. Lately, a promising Arabic QR approach was proposed by Mohtarami et al. (2016) who made use of text similarities at both sentence and word level based on word embeddings. The similarities were computed between new and previous question, and between the new question and the answer related to the previous question. A tree-kernel-based classifier was used in (Barrón-Cedeno et al., 2016) where authors relied on supervised and unsupervised models that operated both at sentence and chunk levels for parse tree based representations. A supervised learning approach was employed in (Malhas et al., 2016) where learning-to-rank models were trained over word2vec features and covariance word embedding features derived from the training data. More recently, the given task was addressed by Romeo et al. (2017) using advanced Arabic text representations built by applying tree kernels to constituency parse trees along with word embeddings and different text similarities. A selection model based on an attention mechanism in an LSTM network was integrated to identify the most important text pieces and then filter out noisy subtrees from the question syntactic trees.

3.5 Conclusion

In this Chapter, we have reviewed the main related work on PR and QR in the context of QA and we have discussed the different models applied to solve these crucial and challenging problems in open domain QA and cQA. Our literature review reveals the widespread attention drawn to QA as well as the abundance of work on the given tasks which reflect their criticality. We devote the next Chapters to presenting out technical contributions.

Chapter 4

Combining multiple features for passage retrieval in open domain QA

4.1 Introduction and motivations

Basically, a common QAS involves a four-modules pipeline (Tellex et al., 2003) namely, question analysis, document retrieval, passage retrieval and answer extraction, where each module has to deal with different challenges. Chiefly, Passage Retrieval (PR) is advocated as the key component of a typical QAS (Krikon et al., 2012) since it allows to reduce the search space from a vast collection of documents to a fixed number of passages. PR is an Information Retrieval (IR) application that returns snippet of texts named passages which are relevant to the user query rather than returning a full list of documents. Obviously, a correct answer to a posted question can be found only when it already exists in one of the retrieved passages. In addition, it has been proved that the performance of the PR module significantly affects that of the whole system (Tellex et al., 2003). Hence, several approaches have been developed for the purpose of PR in order to improve the performance of the QA task, as shown in the previous Chapter. However, most proposed methods are no more than simple adaptations of classical document retrieval engines which are not especially devoted to QA (Buscaldi et al., 2010) and consequently cannot ensure high passage relevance.

As a matter of fact, the task of determining not only a correct but also a relevant answer to a human natural language question over a sizable document collection is far from being easy, and still requires a non-trivial endeavor. Most existing QASs were developed for closed domains with limited capabilities and were unable to deliver correct answers to all given questions from a huge repository even though this later contains the correct answers. Thus, we suppose that if

we improve the PR engine of QASs by increasing the number of relevant passages, we could probably enhance the whole performance and increase the chance of obtaining a relevant answer. Moreover, ranking passages is also deemed to be a laborious subtask at the end of the PR module, which aims to re-rank the retrieved passages such that the most relevant ones appear first.

In order to enhance the performance of existing QASs and ensure the relevance of the returned passages, we propose a new PR and ranking approach for an open domain QAS. Specifically, our contributions include a novel n-gram based method to retrieve passages for QA based on the degree of closeness or dispersion of the n-gram words of the question in the passage (Faiz & Othman, 2019). Another contribution is to better re-rank the passages using a Ranking SVM model that combines a set of text similarity measures which constitute the features (Othman & Faiz, 2016b). These latter include our proposed n-gram measure as well as other lexical, syntactic and semantic features which have already shown promise in the Semantic Textual Similarity task (STS) at *SEM 2013 (Buscaldi et al., 2013). We intend to automatically return the top ranking passage as the most relevant response to a given question stated by the user.

The rest of this Chapter is structured as follows. In Section 4.2, we introduce our approach and we detail its different steps. In Section 4.3, we move on to the description of our experimental study carried out to validate our work by means of the CLEF dataset. We compare our results with those of similar solutions performing the same task and we discuss our findings. Finally, in Section 4.4 concluding remarks are outlined.

4.2 Description of the proposed PaROD approach

The intuition behind our approach is to reduce the search space relying on n-gram structures by retrieving the top 10 passages that are most likely to fit the user's question (Othman & Faiz, 2016b). The number of retrieved passages is set at 10 as most of existing PR systems take values close to 10 as a happy medium between a big and small number. However, n-gram seems not enough to guarantee high relevance since it only relies on a simple dependency between terms. Thus, we tend to better rank the retrieved passages using a Ranking SVM model which combines additional lexical and semantic similarity measures in order to output the top ranking one, as the most relevant passage to the input question. In this Section, we describe our approach which is basically composed of three main components: question preprocessing, PR and passage ranking. Figure 4.1 depicted the overall architecture of the proposed Passage Retrieval for Open Domain QA approach referred to as PaROD. In what follows, we detail its different components mainly, passage extraction and passage ranking concerned with our contributions.

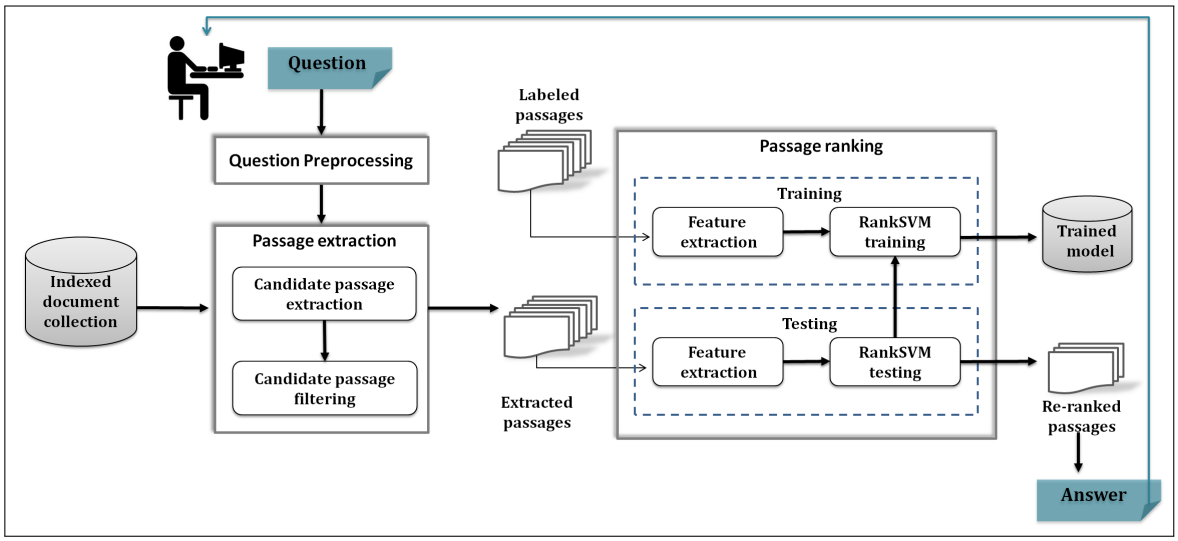


Figure 4.1 – Global architecture of the PaROD approach

4.2.1 Question preprocessing

Preprocessing is a crucial step in NLP tasks to assess and improve the quality of text data in order to ensure the reliability and validity of the statistical analysis. Basically, it intends to skip useless information and extract the terms that may be valuable in the retrieval process. The output of our question preprocessing module should be a formal query generated by preprocessing and analyzing the question posted by the user and extracting the useful terms. This query is obtained by following a few steps: The question is first cleaned by replacing the accented characters with non accented ones and eliminating some special characters such as {$, £, &, §}, punctuation marks as well as question words such as “who”, “what” and “which” which are not useful for the search and therefore they are not included in the query. Letters are lowercased while dates are replaced by the token *date* and numerical digits are normalized to the token *num*.

Then, the question text is chopped up into tokens using a lexical analysis, where a token can be defined as an instance of a sequence of characters in a document, grouped together as a useful semantic unit for text processing that should be not only linguistically significant, but also methodologically useful.

We therefore remove the stop words which refer to extremely common words such as “the”, “is”, “at” which are likely to be of negligible value in the process of selecting documents matching a user need and thus it is better to filter them out from the vocabulary entirely. Thereafter, the terms will be sorted by collection frequency taking the most frequent ones.

We emphasize that stemming is recommended for question preprocessing in order not to loose passages containing question stems. We remind that stemming is a popular NLP technique

that aims to reduce words to their root form by returning just one base form for them according to their morphological variants, including the removal of derivational affixes. Thus, at the end of the question preprocessing module, we obtain a query formally defined as follows:

$$Q = \left\{t_1, t_2, ..., t_q\right\} \tag{4.1}$$

where t denotes a separate term of the query Q and q represents the number of query terms.

4.2.2 Passage extraction

The passage extraction module is a key component in our approach and it consists of two main phases: In the first one named *Candidate Passage Extraction*, we extract all passages containing at least one of the question terms. Then, in the second phase named *Relevant Passage Extraction*, we filter the extracted candidate passages in order to keep the potentially relevant ones using n-gram structure.

4.2.2.1 Candidate Passage Extraction:

The extraction of candidate passages entails three main steps described below, namely collection indexing, term weighting and candidate passage filtering.

Collection Indexing: The document collection should be in advance splitted into paragraphs called passages. In order to index the collection, for each passage, we store its id, number and text and other related information such as its document name. Note that the same question pre-processing steps were applied to passages to obtain a set of indexed passages. We denote by $T(P) = \left\{t_1, t_2, ..., t_p\right\}$ the sequence of p terms obtained by pre-processing a passage P. Subsequently, we give a frequency to each passage term in order to calculate the maximum frequency and the term weights. Once the passages are indexed, the terms are stored in the inverted index as shown in Table 4.1.

Table 4.1 – An excerpt of the inverted index

term Id	term	term frequency in all passages	number of passages containing the term	passages num °	frequency term/passage
1	variable	3	3	1509 1514 1723	1 1 1
2	include	5	4	65 198 818 1319	1 2 1 1
..					

The query and the passage will be modeled as vectors containing the terms as well as their frequencies as given below:

$$\overrightarrow{Q}\begin{pmatrix} t_1 & tf_1 \\ t_2 & tf_2 \\ .. & .. \\ t_q & tf_q \end{pmatrix}\overrightarrow{P}\begin{pmatrix} t_1 & tf_1 \\ t_2 & tf_2 \\ .. & .. \\ t_p & tf_p \end{pmatrix}$$

where t denotes a separate term and tf represents its frequency in the query or the passage.

Term Weighting: In order to calculate the weight of the query terms, we test two formulas. The first one is a variant of the standard formula based on the tf and idf criteria:

$$w(t_i, q) = (\frac{tf(t_i, q)}{maxtf(t_j, q)}) \times \log(\frac{N}{n(t_i)}) \tag{4.2}$$

where $tf(t_i, q)$ is the frequency of the term i in the query and $maxtf(t_j, q)$ is the maximum frequency of the terms in the query. This ratio helps to favor the most frequent terms. Indeed, it is equal to 1 (maximum value) for the most frequent term and equal to $1/maxtf(t, q)$ (minimum value) for the least frequent one. The second argument is a ratio between the total number N of passages in the collection and $n(t_i)$ is the number of passages containing the term i. This ratio helps to favor the most discriminated terms in the passages. The second formula 4.3 tested is the one used in (Correa et al., 2010b):

$$w(t_i, q) = 1 - \frac{log(n(t_i))}{1 + logN} \tag{4.3}$$

This formula does not take into account the frequency of words but it only considers their discriminating power between passages. Thus, a term found in a single passage will have a maximum weight as the ratio value in the formula is low. Candidate passages are those that contain at least one of the query terms. To identify them, we just need to look for the query terms in the inverted index, where for each term the list of related passages is recorded, and take the intersection of these passages. The candidate passages are defined as follows:

$$Pc = \{P_1, P_2, ..., P_n\} \tag{4.4}$$

where P_i is a candidate passage and n is the number of candidate passages. Note that the weight of the candidate passage terms is calculated by the same way as the query terms.

Candidate Passage Filtering: In order to filter the candidate passages, we calculate the similarity between each one and the question using the similarity measure given in formula 4.5 that

only considers words in common between the query and the passage as follows:

$$s(p,q) = \frac{\sum_{t_i \in p \cap q} w(t_i, q)}{\sum_{t_i \in q} w(t_i, q)} \tag{4.5}$$

The candidate passages are then ranked according to their similarity scores and their number (n) will be reduced to (nb). More concretely, we only keep the top nb passages having the higher similarity scores. We denote by $Cp(q) = \{P_1, P_2, ..., P_{nb}\}$ the set of nb returned filtered passages. where $s(p_1, q) \geq s(p_2, q) \geq ... \geq s(p_{nb}, q)$.

Recall that we attempt to reduce the overall system complexity in terms of time and space and allow for a deeper analysis which was not possible before because of the massive size of the collection. Whence, the number nb should be fixed trying to find a happy medium between a big and a small number. On one hand, a big nb does not meet the goal mentioned above, but it has the advantage of keeping passages that may be semantically relevant but are misclassified as they share very few words with the query. On the other hand, a small number can reduce the system complexity but with a strong chance of ignoring some passages. In view of this, we propose to set the number nb to 100 as an average number of candidate passages.

4.2.2.2 Relevant Passage Extraction:

The extracted passages returned by the previous phase, until then, have only a few question words. However, we believe that this criterion is not good enough to assess the passage relevance. Thus, we propose to go beyond a simple verification of word occurrences by exploiting other selection criterion such as the presence of word sequences, their length and their dependence. To this end, we resort to n-gram technique which can not only deal with a massive amount of data but also with its heterogeneity and it is further independent of language. Our methodology for extracting relevant passages is composed of the following parts:

N-gram generation: We just focus on the common n-grams between the question and the passage. Thus, the common terms between a question and a passage are first identified and then, their corresponding n-grams are derived. The vector $\overrightarrow{Tc}$ of common terms between the question and the passage is built by browsing through the terms of the question and checking for each of them if it is also a term of the passage to add it in the vector. This latter is defined by:

$$\overrightarrow{Tc} \begin{pmatrix} t_1 & p_1 Q & [p_{11}, .., p_{1m}] \\ t_2 & p_2 Q & [p_{21}, .., p_{2m}] \\ .. & .. & \\ t_n & p_n Q & [p_{n1}, .., p_{nm}] \end{pmatrix}$$

where t_i is the term i in common between the question and the passage and i={1..n}. n denotes the number of the question terms, p_iQ denotes the position of the term i in the question, p_{ij} is the position j of the term i in the passage and j={1..m}. m is the number of terms in the passage. These positions will be useful to construct the n-grams of the question and the passage. Thereafter, the n-grams of the question are constructed by browsing the vector $\overrightarrow{Tc}$ and grouping the terms having successive positions in the question. These question n-grams are defined by a vector $\overrightarrow{NGQ}$. Similarly, the n-grams of the passage are constructed and defined by a vector $\overrightarrow{NGP}$ using the vector $\overrightarrow{Tc}$.

$$\overrightarrow{NGQ}\begin{pmatrix} ngQ1 \\ ngQ2 \\ .. \\ ngQq \end{pmatrix} \overrightarrow{NGP}\begin{pmatrix} ngP_1 \\ ngP_2 \\ .. \\ ngP_p \end{pmatrix}$$

where q and p denote the number of n-grams in the question and the passage, respectively.

This straightforward intuition for the construction of the n-grams of the question and the passage are given in the pseudo-algorithms 1 and 2 (Faiz & Othman, 2019).

Algorithm 1 An algorithm for constructing the question n-grams

Input:
The set of common terms and their positions in the question: {(t_1, p_1),. . . , (t_n, p_n)}
Output:
The number of n-grams of the question: q
The set of n-grams of the question : {ng_1, . . . , ng_q}

q=1 ; ng(q)=t_1;
for $i = 2 \rightarrow n$ **do**
  if $p_i = p_{i-1} + 1$ **then**
    // check if t_{i-1} and t_i is a bi-gram in a question
    ng(q) ← ng(q) + t_i; // concatenate t_i with the n-gram q
  else
    q++; // increment the number of n-grams
    ng(q) ← t_i ; // add t_i to the new n-gram q
  end if
end for

Algorithm 2 An algorithm for constructing a passage n-grams

Input:

The set of common terms and their positions in the passage: {(t_1, [p_{11},...,p_{1m}])],..., (t_n, [p_{n1},...,p_{nm}])}

Output:

The set of n-grams of a passage : {$ng_1, \ldots, ng_p$}

p=1 ; ng(p)=t_1;
for $i = 2 \rightarrow n$ **do**
  if ($p_{i^*} = p_{i^*-1} + 1$) **then**
    // * represents any number from 1 to m
    // check if t_{i-1} and t_i is a bi-gram in the passage
    ng(p) $\leftarrow$ ng(p) + t_i; // concatenate t_i with the n-gram p
  else
    p++; // increment the number of n-grams
    ng(p) $\leftarrow$ t_i ; // add t_i to the new n-gram p
  end if
end for

N-gram Weighting: The weight of each n-gram of the question is calculated on the basis of its length and the sum of its term weights according to formula 4.6:

$$w(ngQ) = l \times \sum_{t_i \in terms(ngQ)} w(t_i, q) \tag{4.6}$$

where l is number of terms contained in the n-gram (ngQ).

Indeed, the multiplication of the weights sum by the n-gram length can foster adjacent words over the independent ones in the similarity calculation. We believe that grouped terms are more significant and less ambiguous than separate ones. Therefore, a term that belongs to an n-gram should have a greater weight than an independent one. The weights of terms belonging to a common n-gram in question and passage are then re-inforced by a factor proportional to n; the length of the n-gram l.

As for the passage n-grams, they are weighted regarding their similarity degree with those of the question. We give a cumulative weight to the passage by browsing through the question n-grams and at each question n-gram either its full weight or a lower one is added to the passage weight. In other words, if a question n-gram occurs in the passage, its whole weight will be added to the total weight, while if it is divided in the passage into smaller subset of n-grams, called sub-n-grams, a lower weight will be added to the cumulative weight. This lower weight should be fixed according to the number of sub-n-grams. We emphasize that three possible cases

may arise:

- Case 1: The n-gram of the query is one of the passage n-grams: $ngQ_i \exists ngP_j, ngP_j \in NGP$
- Case 2: The query n-gram combines several passage n-grams; it is included in the union of a subset of passage n-grams: $ngPP \subset NGP$, $ngQi \subset (\bigcup ngP_j \in ngPP)$ where $ngPP$ denotes some subset of NGP.
- Case 3: The query n-gram is a sub-n-grams of some passage n-gram: $ngQ_i \in ngP_j, ngP_j \in NGP$

Let w be the weight to add to the passage when we browse through the question n-grams ngQ. In the cases 1 and 3, ngQ exists in the passage, so the additional weight w is calculated using the formula 4.7:

$$w(ngP) = w(ngQ) = l \times \sum_{t_i \in terms(ngQ)} w(t_i, q) \tag{4.7}$$

where l denotes the length of the n-gram ngQ and $w(t_i, q)$ is the weight of its term t_i. In the case 2, ngQ is divided into sub n-grams in the passage, let sng be the number of these sub n-grams. In this case, the additional weight w is computed using the formula 4.8:

$$w(ngP) = \frac{w(ngQ)}{sng} = \frac{l}{sng} \times \sum_{t_i \in terms(ngQ)} w(t_i, q) \tag{4.8}$$

Passim Similarity Measure: Our proposed passage similarity measure referred to as *Passim* between a passage and a question is no more than the ratio between the weight of the passage and that of the question (Faiz & Othman, 2019). The passages are ordered according to this measure scores in order to return those having the highest values. The final passage weight is a sum of partial weights calculated at each step to be added to the total passage weight. The next step corresponds to the verification of the occurrence of an n-gram of the question into the passage and the underlying weight is given by the following formula 4.9:

$$w(P) = \sum_{i=1}^{q} \frac{l_i}{sng_i} \times \sum_{t \in terms(ngQ_i)} w(t, q) \tag{4.9}$$

where q is the number of the question n-grams. The weight of a passage is calculated including sng_i which denotes the number of sub-n-grams in the passage corresponding to the question n-gram i. The number sng_i is calculated during the constitution of the n-grams of the question and the passage using the pseudo-algorithm 3. The basic intuition behind this latter is to browse through the common terms between the question and the passage where n denotes their number,

and for each question n-gram, which can be thought as terms having successive positions in the question, we verify if it also represents an n-gram in the passage, otherwise, we check if the given question n-gram is divided into sub-n-grams in the passage and we calculate their number. Note that pq_i denotes the position of the ith term in the question while pp_{im} denotes the mth position of the ith term in the passage. As for the weight of the question, it is calculated according to formula 4.10:

$$w(Q) = l(Q) \times \sum_{t_i \in (Q)} w(t_i, q) \tag{4.10}$$

where $l(Q)$ is the number of the question terms and $w(t_i, q)$ is the weight of a question term. $w(Q)$ is the same as the weight of an n-gram when the n-gram NGQ is the question. $k = l(Q)$ represents the number of the question terms and $sng = 1$ since all terms are grouped together and form a uni-gram.

The similarity between a question Q and some passage P is defined as the ratio between $w(P)$, the weight of n-grams in the passage, and $w(Q)$, the weight of n-grams in the question as given in 4.11:

$$Passim(p, q) = \frac{w(P)}{w(Q)} \tag{4.11}$$

Obviously, this similarity is maximum when all the terms of the question are grouped in the passage.

In a nutshell, our strategy for retrieving passages is different from the existing ones based on n-grams. Firstly, the given process of n-gram extraction extract only common n-grams between the question and the passages with different sizes, instead of extracting all n-grams for all n possible gram values of the question and the passage, as in (Buscaldi et al., 2010) and (Correa et al., 2010b), or all the n-grams of size n, as in (Radev et al., 2005). Thus, no additional step is required to select common n-grams from all the extracted ones. Secondly, for the weight of n-grams, both the sum of the terms weight and their lengths are considered like in (Radev et al., 2005), while (Correa et al., 2010b) and (Buscaldi et al., 2010) consider only the sum of the terms. Thirdly, the passage similarity measure is calculated from the weights of the question n-grams. A total passage weight is built by browsing the question n-grams, keeping the weight of an n-gram if it is totally included in the passage and reducing the weight of an n-gram if it is divided into smaller n-grams in the passage.

Algorithm 3 An algorithm for calculating the sub-n-grams related to an n-gram of the question in the passage

Input:
The set of terms and their positions in the question and in the passage {(t_1,pq_1 [pp_{11},...,pp_{1m}]),..., (t_n,pq_n [pp_{n1},..., pp_{nm}]) }
Output:
The set of n-grams of the question {ngq_1, ..., ngq_q}
The number of n-grams of the question q
The number of n-grams in each sub-n-gram of the question {sng_1,...,sng_q}
The set of n-grams of the passage {ngp_1, ..., ngp_p}

q=1 ; ngq(q)=t_1 ; sng(q) = 1 ;
p=1 ; ngp(p)=t_1 ;
for $i = 2 \rightarrow n$ **do**
  if (pq_i= pq_{i-1}+1) and (pp_{i*} = pp_{i-1*} + 1) **then**
    ngq(q) ← ngq(q) + t_i ; // concatenate t_i with the n-gram q
    ngp(p) ← ngp(p) + t_i ; // concatenate t_i with the n-gram p
  else
    if (not(pq_i= pq_{i-1}+1) and (pp_{i*} = pp_{i-1*} + 1)) **then**
      q++ ; // increment the number of n-grams of the question
      ngq(q) ← t_i ; // add t_i to the new n-gram q
      ngp(p) ← ngp(p) + t_i ;// concatenate t_i with the n-gram p
      sng(q) ← 1 ;
    end if
    if ((pq_i= pq_{i-1}+1) and not((pp_{i*} = pp_{i-1*} + 1))) **then**
      p++ ; // increment the number of passage n-grams
      ngp(p) ← t_i ; // add t_i to the new n-gram p
      ngp(q) ← ngp(q) + t_i ; // concatenate t_i with the n-gram q
      sng(q) ++ ;
    end if
    if (not (pq_i= pq_{i-1}+1) and not((pp_{i*} = pp_{i-1*} + 1))) **then**
      q++ ; // increment the number of the question n-grams
      ngq(q) ← t_i ; // add t_i to the new n-gram q
      p++ ; // increment the number the passage n-grams
      ngp(p) ← t_i ; // add t_i to the new n-gram p
      sng(q) ← 1 ;
    end if
  end if
end for

In order to elucidate the idea of *Passim* measure we propose a simple example 4.1, where we consider the following terms of a question Q and those of a passage P:

Example 4.1. *Q(terms)= trade,* ***ammonium****,* ***nitrate****,* ***fertilizers****, hampered,* ***European****,* ***Economic****,* ***Community****.*

P(terms)= ***ammonium****,* ***nitrate****, essential, ingredient, variety, products, some, intended, use,* ***fertilizers****, others, explosives, reason, divergencies, national, provisions, classification, content,* ***European****,* ***Economic****,* ***Community****, regulations, controlling, marketing .*

The corresponding vector $\overrightarrow{Tc}$ of common terms between the question and passage is set to:

$\overrightarrow{Tc(Q,P)}$*= ammonium, nitrate, fertilizers, European, Economic, Community.*

From this latter we can derive:

$\overrightarrow{NGQ(Q)}$ *=[ammonium nitrate fertilizers][European Economic Community]*

and $\overrightarrow{NGP(P)}$ *=[ammonium nitrate][fertilizers][European Economic Community].*

In this example, $\overrightarrow{NGQ}$ is composed of two n-grams so we have two partial passage weights to calculate. The first question n-gram is divided into two sub n-grams in the passage so, sng equals 2 while the second one is exactly equal to a passage n-gram so, sng equals 1. Thus, given 1.766 and 1.524 the term weights of ngQ_1 and ngQ_2 respectively, $w_1(P) = \frac{l(ngQ_1)}{sng_1} \times \sum_{t \in terms(ngQ_1)} w(t,q) = (3/2) \times 1.766$ while $w_2(P) = \frac{l(ngQ_2)}{sng_2} \times \sum_{t \in terms(ngQ_2)} w(t,q) = (3/1) \times 1.524$. Therefore, the total passage weight will be equal to the sum of $w_1(P)$ and $w_2(P)$, while the weight of the question will be set to: $w(Q) = l(Q) \times \sum_{t_i \in (Q)} w(t_i,q) = 8 \times 4.924$ where 4.924 is the result of the sum of query terms weight.

4.2.3 Passage ranking

Since the n-gram technique can only ensure simple dependencies between terms, it cannot guarantee that the retrieved passages are highly relevant. We accordingly suggest integrating other powerful similarity measures combined by means of a Ranking SVM model. This model combines different text similarity measures that constitute the features (Othman & Faiz, 2016a). We emphasize that the passage ranking model consists of two phases: training and testing, as depicted in Figure 4.1.

In both phases, the different similarity measures are calculated for each passage and then these latter are entered into the RankSVM which will re-rank the passages given their feature values. Only the passage ranked first by our model will be returned by the system as the most relevant answer to a given user's question. During the first phase, a set of annotated passages

entered in the passage re-ranking model where each passage is labeled either 1 (right) or -1 (wrong), while in the testing phase, the passages are not labeled as they are those extracted by our PR module. In what follows, we present the different used features and we give an overview on the RankSVM model.

4.2.3.2 Feature Extraction:

In addition to *Passim*, we integrated in our ranking model other features which have shown success in the Semantic Textual Similarity task (Buscaldi et al., 2013)(STS) at *SEM 2013 which aims at determining the degree of similarity between pairs of text sentences. Among the introduced features, we pick out WordNet-based Conceptual Similarity, Named Entity Overlap, Edit distance. We have adapted these features to the context of QA, where the sentence pairs become pairs of passage-question. Basically, what is expected from the additional features retained for ranking is to take into account the semantic relationships, the syntactic structures and the named entities containing in passages in order to improve the ranking, ensure more passage relevance and enhance the n-gram-based passage extraction module. Note that the more we add features, the higher the program complexity is. Hence, we have mainly resorted to syntactic and semantic features to ensure answer relevance (Keikha et al., 2014) since at this stage, similarity measures based on term frequencies are insufficient to ensure high passage relevance. The given used features are briefly described below.

WordNet-based Conceptual Similarity: The question q and a passage p are first analysed in order to extract all the corresponding WordNet synsets. For each one, we keep only noun synsets and group them respectively into the set of synsets of the question called C_q and that of the passage named C_p. If the synsets belong to another POS categories such as verb, adjective, pronoun, we seek their derivational related forms to obtain a related noun synset and we add it in the corresponding set of synsets.

Example 4.2. *For example, the word "playing" in WordNet is associated to synset* (*v*)*play*#2*, which has two derivationally related forms: synsets* (*n*)*play*#5 *and* (*n*)*play*#6*. These latter are the synsets that should be added to the synset set.*

Given C_p and C_q the sets of concepts contained in a passage p and the question q, with $|C_p| \geq |C_q|$, WordNet-based Conceptual Similarity between p and q is set to:

$$ss(p,q) = \frac{\sum_{c_1 \in C_p} \underset{c_2 \in C_q}{maxs}(c_1, c_{2,})}{|C_p|} \tag{4.12}$$

where $s(c_1, c_2)$ is a conceptual similarity measure calculated using a variation of the Wu-Palmer formula (Z. Wu & Palmer, 1994), called ProxiGenea (genealogical proximity) introduced by Dudognon et al. (2010). We will employ the third version of this measure named ProxiGenea since it has outperformed the other ones in the Semantic Textual Similarity task (Buscaldi et al., 2012). This latter is defined as follows:

$$pg_3(c_1, c_2) = \frac{1}{1 + d(c_1) + d(c_2) - 2d(c_0)} \tag{4.13}$$

where c_0 denotes the most specific concept that is present both in the synset path of c_1 and c_2 and d denotes the function returning the depth of a given concept which constitutes the number of nodes between a concept and the root in the WordNet taxonomy (see Figure 4.2 for details).

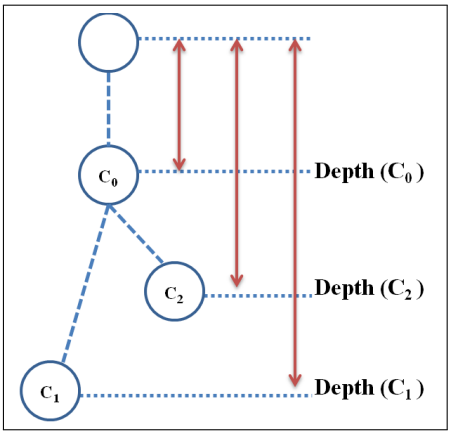


Figure 4.2 – Illustration of depth calculation

Example 4.3. *In order to elucidate the principle of this measure, we suggest a simple example where we propose to calculate WordNet-based conceptual similarity between the following sentences:*

- *Sentence 1: After the tornado hit the town, there was little left standing.*
- *Sentence 2: The city was partially evacuated after the storm.*

The two sentences are analyzed and pos tagged to extract all the corresponding noun WordNet synsets. For each pair of nouns, we calculate the ProxiGenea measure as shown in Table 4.2 below. For instance, the terms town and city have similar meaning so they are assigned a high proxigenia value equal 0.900. In our example, the WordNet-based conceptual similarity value

will be set to:

$$ss(p,q) = \frac{\sum_{c_1 \in C_p} \underset{c_2 \in C_q}{maxs}(c_1, c_{2,})}{|C_p|} = \frac{0.900 + 1.000 + 0.571 + 0.857}{4}$$

Table 4.2 – Proxigenia similarity between two concepts

	After /IN	**the** /DT	**tornado** /NN	**hit** /VBD	**the** /DT	**town** /NN	**there** /EX	**was** /VBD	**little** /JJ	**left** /NN	**standing** /VBG
The /DT	-	-	-	-	-	-	-	-	-	-	-
city /NN	-	-	0.316	-	-	0.900	-	-	-	0.737	-
was /VBD	-	-	-	0.500	-	-	-	1.000	-	-	0.800
partially /RB	-	-	-	-	-	-	-	-	-	-	-
evacuated /VBN	-	-	-	0.571	-	-	-	0.400	-	-	0.500
after /IN	-	-	-	-	-	-	-	-	-	-	-
the /DT	-	-	-	-	-	-	-	-	-	0.528	-
storm /NN	-	-	0.857	-	-	0.375	-	-	-	-	-

Named Entity Overlap: A Named Entity Recognizer(NER) is used such as that implemented by (Finkel et al., 2005), which is composed by 7 classes: Organization, Person, Money, Time, Location, Percent, Date. Thereafter, a per-class overlap measure is computed considering the class of each named entity. For instance, "Tunis" as a Location does not correspond to "Tunis" as an Organization. This similarity measure is calculated as:

$$O_{NER}(p,q) = \frac{2 * |N_p \cap N_q|}{|N_p| + |N_q|} \tag{4.14}$$

where N_p and N_q are the sets of named entities detected in p and q.

Edit distance: Edit distance is a similarity measure that quantifies how dissimilar two strings or text fragments are by calculating the minimum number of operations needed to transform one string into the other. Numerous variants of edit distance using different operations exist. One of the most common one, that we have applied, is called Levenshtein distance. The edit distance is defined as follows:

$$sim_{ED}(p,q) = 1 - \frac{Lev(p,q)}{max(|p|,|q|)} \tag{4.15}$$

where $Lev(p,q)$ is the Levenshtein distance between the passage and the question.

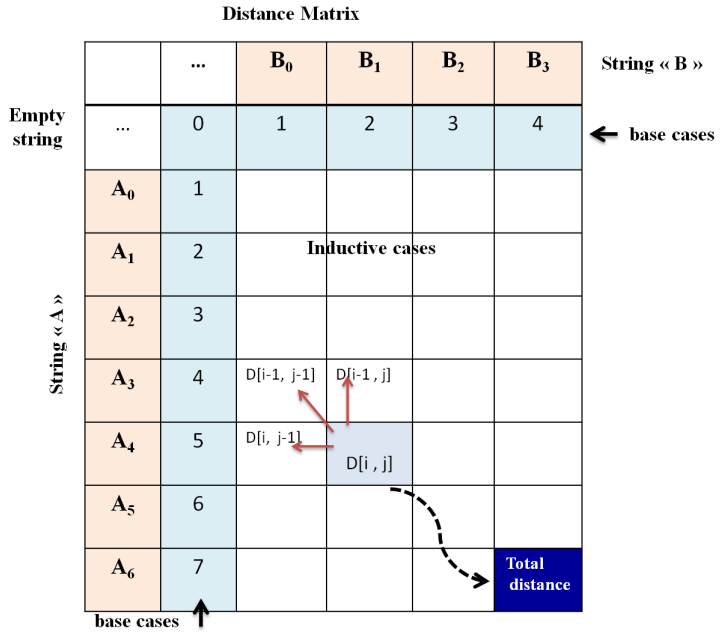


Figure 4.3 – Distance matrix for the edit distance

Example 4.4. *We propose to calculate the edit distance between two simple words SPARTAN and PART. To this end, we suggest to create a matrix where we represent all possible combinations of one set of entities with another as shown in Figure 4.3 where* $D_{[i,j]}$ *is the edit distance between the substring of a string A of length i and the substring of B of length j. Note that the base case edit distance values denote just the length of the sub-strings while the inductive cases represent the minimum edit distance between two strings. Two possibilities may arise, depending on the last characters in the two strings. If the last characters are equal, then* $D_{[i,j]}$ *will be equal to* $D_{[i-1,j-1]}$*. Otherwise, if the characters are not equal, then* $D_{[i,j]}$ *will be equal to* $1 + min(D_{[i,j-1]}, D_{[i-1,j]}, D_{[i-1,j-1]})$*. The steps of the transformation of the term SPARTAN into PART are detailed in Table 4.3.*

Table 4.3 – Transformation steps of ”SPARTAN” into ”PART”

step	comparison	edit necessary	total editing
1	“S” to “ ”	delete: +1 edit	“S” to “ ” in edit
2	“P” to “P”	no edits necessary!	“SP” to “P” in 1 edit
3	“A” to “A”	no edits necessary!	“SPA” to “PA” in 1 edit
4	“R” to “R”	no edits necessary!	“SPAR” to “PAR” in 1 edit
5	“T” to “T”	no edits necessary!	“SPART” to “PART” in 1 edit
6	“A” to “T”	delete: +1 edit	“SPARTA” to “PART“ in 2 edits
7	“N” to “T”	delete: +1 edit	“SPARTAN” to “PART” in 3 edits

The last case of the matrix reveals the minimum number of edits needed to transform the first word into the other. We just need to climb following the arrows in the matrix to discover the different edits which are presented in the above Table. Thus, the Edit distance value corresponding to our example will be set to:

$$sim_{ED}(A, B) = 1 - \frac{Lev(p, q)}{max(|p|, |q|)} = 1 - (\frac{3}{7})$$

where the maximum length of the two string equals 7.

4.2.4 Ranking SVM model

This subsection briefly point out the main idea of RankSVM which we have applied in our approach. It is worth noting that in our case, the RankSVM model receives a file containing a list of the k passages retrieved by the PR module and their corresponding values of the different features. The output of RankSVM consists of a list of scores of the candidate passages. Each score is a fraction between 0 and 1. The final output of our system should be the passage having the highest score. We set a threshold value for the final score to be 0.15 as it has been chosen by many authors for the final ranking result. If the highest score value exceeds 0.15, we answer the question. Otherwise, we choose not to return any answer to the question rather than giving a wrong answer.

Ranking Support Vector Machine, referred to as RankSVM, is a ranking version of the SVM model introduced by Herbrich et al. (1999) to solve the ranking problem in a supervised manner. The underlining idea behind this popular machine learning method is to transform the ranking problem into pairwise classification and then learn a prediction ranking function using the SVM intuition. By and large, the algorithm works in two passes. In the first one, it classifies the different pairs of objects while in the second pass, it ranks test queries.

The problem of extending an SVM to RankSVM is formally defined as follows: Assume that there exists an input space $X \in R^n$, where $x \in X$ denotes an object and n represents the number of features. Further assume that there exists an output space of ranks or categories denoted by labels $Y = \{r_1, r_2, .., r_q\}$ where q is the number of ranks. We suppose that there exists a total order between the ranks $r_q \succ r_{q-1} \succ .. \ r_1$ where $\succ$ represents a preference relationship. Note that there is a set of ranking functions $f \in F$ where each one points out the preference relations between instances. For example, if $x_i \succ x_j$, then $f(x_i) \succ f(x_j)$. Now, suppose that we have a set of ranked instances denoted by $S = (x_i; y_i)$ from the space $X \times Y$. The objective is to determine the best function f from F that minimizes a given loss function. This rank learning problem has been formalized by Herbrich et al. (1999) to turn the rank problem into a binary classification problem. We further assume that w is a weight vector that corresponds to the ranking linear function: $f_w(x) = (w, x)$ which can score and rank the instances.

To elucidate the main idea of this problem, we suggest the following example 4.5 which explains how to adapt the SVM classifier for pairwise classification to the ranking problem.

In fact, the differences between two instances at different levels x_i and x_j, in the same group are expressed as new feature vectors defined as $x_i - x_j$, (e.g., $x_1 - x_2$, $x_1 - x_3$ and $x_3 - x_2$). Thus, the original training dataset S is turned into a new training dataset S':

Example 4.5. *Suppose that we have two different groups of objects which can represents documents associated with two distinct queries in the feature space. We further suppose that there are three possible ranks: definitely relevant (r3), partially relevant (r2), and irrelevant (r1). The giving problem is illustrated in Figure 4.4. It is noteworthy that only objects belonging to the same group can be comparable such as x_1, x_2 and x_3. The objects can be scored by the weight vector and ranked with the function by projecting them into the vector and sorting them according to the obtained scores.*

As mentioned above, in order to handle this ranking problem with SVM we should transform it into a pairwise classification. Figure 4.5 illustrates the transformation of the ranking problem in Figure 4.4 into a linear SVM classification. Thus, the original training dataset S is turned into a new training dataset S' which is set to:

$$(x_i^1 - x_i^2, z)z = \begin{cases} +1 & y^1 \succ y^2 \\ -1 & y^2 \succ y^1 \end{cases} \tag{4.16}$$

where x^1 and x^2 denote the first and second instances, y^1 and y^2 represent their ranks and z denotes their labels which can be either positive ($z = +1$) or negative ($z = -1$). Note that this transformation of the space is possible thanks to kernel functions (e,g., linear kernel, polynomial kernel, sigmoid kernel).

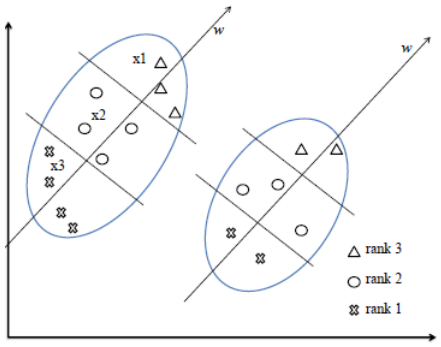


Figure 4.4 – Illustration of the Ranking problem

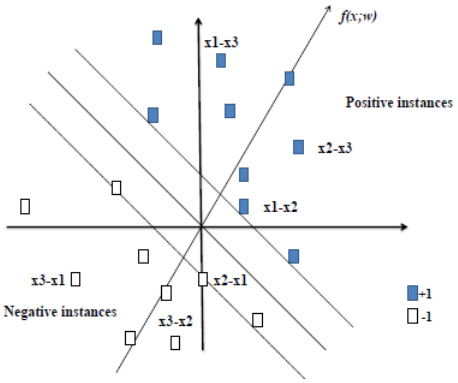


Figure 4.5 – Transformation of the ranking problem to pairwise classification

Back to our example, the new instances resulting from the transformation are: $x_1 - x_2$, $x_1 - x_3$, $x_2 - x_3$, $x_2 - x_1$, $x_3 - x_2$, $x_3 - x_1$. *The labels assigned to these latter are respectively: 1, 1, 1, -1, -1, -1. As shown in Figure 4.4, the instances* $x_1 - x_2$, $x_1 - x_3$ *and* $x_2 - x_3$ *are positive while the other ones are negatives.*

Basically, the learning of RankSVM is formalized as the following Quadratic Programming problem given in equation 4.17:

$$\begin{aligned} min_{w,\xi} \frac{1}{2} \|w\| + C \sum_{i=1}^{m} \xi_i \\ s.t. : y_i(w, x_i^1 - x_i^2) \geq 1 - \xi_i \\ \xi_i \geq 0 \\ i = \{1...m\} \end{aligned} \tag{4.17}$$

where x_i^1 and x_i^2 represent respectively the first and the second instances in a pair of feature vectors for a given query, $||w||$ denotes the margin of the hyperplane, ξ_i is a slack variable, $C \succ 0$ is a coefficient that represents a trade-off between training error and margin and m denotes the number of training instances.

We point out that RankSVM has been successfully applied in the context of IR, in particular to document retrieval (Y. Cao et al., 2006). In our work, we adapt this learning to rank method to passage retrieval in order to re-rank the retrieved passages.

4.3 Experimental evaluation

4.3.1 Datasets

In our experiments, we have tested our approach using the dataset provided in the context of the ResubliQA task proposed by CLEF. For the evaluation of our passage extraction module, we used the dataset provided in the ResPubliQA 2009 exercise (Peñas et al., 2010) which aims to retrieve paragraphs from the test collection to answer a given question picked out from a set of 500 questions. On the other hand, to assess the applicability of the whole approach, we used the resources provided in the ResPubliQA 2010 exercise (Peñas et al., 2010). The objective of this exercise is to return either paragraphs or exact answers as system output to a set of 200 different questions over two test collections. As we work on passages, the given datasets ought to be the most appropriate ones to evaluate our PR approach. We have realized our experiments with different languages to prove that our approach is language independent. Moreover, this choice allows the comparison of our results with those obtained by other systems for the same task. Both datasets of these exercises are briefly described below.

4.3.1.1 Description of the ResPubliQA CLEF collections

ResPubliQA 2009 collection The ResPubliQA 2009 includes JRC-Acquis [1], which is a parallel text corpus currently available in 22 different languages. For each language, the number of documents is almost equal to 23000. For the ResPubliQA task, a subset of the JRC-Acquis collection is used where only corpora for the 8 following languages are considered: Bulgarian, German, English, Spanish, French, Italian, Portuguese, Romanian. For each language, roughly 10700 documents are used. Notice that the documents in the collection are organized in an XML

[1] JRC-Acquis is an approximation of the Acquis Communautaire (AC) which represents the entire body of European Union (EU) laws applied in the the EU Member States, collected between 1950 and 2006, available in https://ec.europa.eu/jrc/en/language-technologies/jrc-acquis

format where each one has a title and is subdivided into paragraphs marked with $\prec p \succ$ tag. Additionally, each document consists of two parts: a header and a body of text divided into sections (text, annex, and signature). The question pool is a set of 500 independent questions grouped in an XML file. These questions fall into five types: factual, definition, reason, purpose and procedure.

ResPubliQA 2010 collection ResPubliQA 2010 collection includes two sets of multilingual parallel documents: a subset of JRC-Acquis and a small portion of the EUROPARL [2] collection. A small subset of the Europarl has been created with parallel documents in all the 9 languages included in the track. The subset consists of approximately 50 parallel documents per language. The question set consists of a pool of 200 more complex questions in 9 different Languages. The question categories used in this track are the following: factoïd, definition, reason-purpose, procedure, opinion and other. The distribution of these questions types in the collection is given in Table 4.4.

Table 4.4 – Distribution of question types

Question type	Total number of questions
DEFINITION	32
FACTOID	35
REASON/PURPOSE	33
PROCEDURE	33
OPINION	33
OTHER	34
Total	200

4.3.1.2 Evaluation of the passage extraction module

We evaluated our passage extraction engine in 3 languages: English, Spanish and French. Note that only few systems have been tested in french due to its ambiguity. Indeed, three participants have performed their tests in french: University Politecnica Valencia SPAIN (Correa et al., 2010b), Synapse Developpment France and LIMSI-CNRS-2 France (Moriceau et al., 2009). Their systems are named, respectively NLEL, SYNA and ILES.

The first experiments are carried out on:

- 10713 documents

[2]Europarl is a collection of the Proceedings of the European Parliament dating back to 1996. It involves text translations of the 11 official languages of the European Union, available in: http://www.europarl.europa.eu/

- 338 questions
- 1388818 passages derived from the document collection
- 100 passages used from those returned by the search model
- 10 passages returned by the n-gram model

In order to set the number of passages returned by the system, we have considered that we want to reduce the number of passages and increase the number of questions with correct answers. Obviously, a high number of passages, even if the number of questions is high too, cannot well judge the performance of the system. In addition, this number is not fixed in the same way in all systems, but most of them take values close to 10. For instance, the system Multitex (Clarke et al., 2000) returns 10 passages, (Ofoghi et al., 2006) and (Verberne et al., 2008) considered 10 passages and (Ofoghi & Yearwood, 2009) considered 10, 15 and 20 passages. We have thereby chosen to return for each question 10 passages.

4.3.1.3 Evaluation of the passage ranking module

In order to verify the applicability of our passage ranking module, we ought to calculate the different text similarity measures for the 10 passages returned by the passage extraction module and integrate them as features using the RankSVM model. Notice that this model consists of two phases: training and testing. In both phases, the features are extracted for each passage and then these latter are entered into the RankSVM to be automatically ranked, and only the top ranking passage will be returned by the system as an output which represents the most promising answer to a given user's question.

During the training phase, a set of labeled passages were entered in the passage ranking model where each passage can be labeled either +1 (right) or -1 (wrong). In fact, for the training, we employed a judgment set manually annotated by CLEF organizers which contains a list of labeled answer candidates. Note that the training receives ResPubliQA2009 english question/answer pairs while in testing, we use the english JRC-Acquis and Europarl collections as well as the question pool proposed in ResPubliQA2010.

The test experiments are carried out on:

- 200 questions
- 10763 documents
- 1404393 passages derived from the document collection

- 10 passages returned by the passage extraction module

4.3.2 Evaluation setup

4.3.2.1 Evaluation of the passage extraction module

To evaluate the performance of our passage extraction module, we developed our approach PaROD in Java using Eclipse environment using the open source system JIRS [3] (JAVA Information Retrieval System) described in (Gómez et al., 2007). We employed the package allowing the management of the database storing the JIRS inverse file in the indexing step. We have developed our processes of indexing and search using those of JIRS and adapting them to our needs.

The passage extraction module takes as input both the document collection and the questions, and returns as output an XML file containing the passages selected to answer a given question picked out by the user. In Figure 4.6, we give an example of XML file returned by our passage extraction engine where the selected question is between two tags: <question id= "0002"> and </question>.

The selected passages are put between two tags $< passage >$ and $< /passage >$ under the tag $< passages >$. Each passage of the list of returned passages is put between $< passage >$ and $< /passage >$ and is identified by an *id* and a number *n* which denotes its number in the document collection.

[3] http://sourceforge.net/projects/jirs/

```
<?xml version="1.0" encoding="utf-8" ?>
<questions>
  <question id="0002">
       Quel est le but de la directive du conseil sur
l'éducation des enfants des travailleurs migrants ?
    <passages>
      <passage id="136291" n="2" doc="jrc31977L0486-fr"
sim="0.6626528">DIRECTIVE DU CONSEIL du 25 juillet 1977 visant à
la scolarisation des enfants des travailleurs migrants
(77/486/CEE)</passage>

      <passage id="689309" n="199" doc="jrc32000D0253-fr"
sim="0.6475536">f) des activités d'éducation et
de...itinérantes;</passage>

      <passage id="689296" n="186" doc="jrc32000D0253-fr"
sim="0.5085404">c) les projets visant à promouvoir ...la ainsi
que des personnes exerçant des professions
itinérantes;</passage>
                          ...
    </passages>
  </question>

</questions>
```

Figure 4.6 – Example of an XML file returned by our PR engine containing the candidate passages that may answer the user's question

In the given exercise, CLEF has produced 2 baselines [4] to judge the correctness of the passages. 11 teams have participated in this task. Each participant submitted an XML file containing for each question the associated passage returned by his system. These answers were evaluated by the campaign organizers to generate an XML file containing judgments for all questions, where a passage identified by a p_id and a doc_id of the $< answer >$, is either correct if the value of the attribute judgment = "CORRECT" or incorrect if the value of the attribute judgment ="INCORRECT".

We recovered the XML files judged for the participants and the XML files judged by both baselines in order to combine all the questions that have correct answers in one XML file. From this latter we generated two files, one contains the questions which will be the system input, and the other contains the correct answers for each question.

4.3.2.2 Evaluation of the passage ranking module

In order to rank the passages, we resorted to the open source SVM^{light}[5], which is an implementation of Vapnik's SVMs for the problems of pattern recognition, regression and learning a

[4] A baseline is an evaluation of the QA exercise, made by the organizers of the competition and which serves as a basis for comparing the results of the participants. The answers to questions for the baselines are obtained by applying a search model and the difference between the two baselines consists of using a stemming technology.

[5] http://svmlight.joachims.org/

ranking function. Note that the optimization algorithms employed in SVM light are presented in (Joachims, 2002). Endowed with scalable memory requirements, this algorithm is able to efficiently handle problems with thousands of support vectors.

Recall that RankSVM receives as input two files (train and test) with similar structures, where each line represents the different values of the features corresponding to one passage. Each feature value denotes the similarity between a passage and the question having a given *id*. Each participant in the paragraph selection task has submitted an XML file containing for each question the associated passage returned by his system. These answers were evaluated by the campaign organizers to generate an XML file containing judgments for all questions. A given passage that includes the right answer information is considered as a correct example and incorrect one otherwise. We have combined the different judgement files in one XML file containing for each question the different proposed answers for the participant systems and the related judgements.

The output of the RankSVM model is a list of scores related to the 10 retrieved passages from the passage extraction module. The passage having the highest score rank is then identified given its *id* and returned by the whole system as a relevant answer to the user's question.

We emphasize that most of the features applied in our ranking model are those proposed in the Semantic Textual Similarity task (Buscaldi et al., 2013)(STS) at *SEM 2013, namely WordNet-based Conceptual Similarity, Named Entity Overlap and Edit distance and our N-gram based Similarity. Indeed, the implementation of these features require a set of resources described below.

- **WordNet-based Conceptual Similarity**: This semantic feature is based on WordNet which is an online lexical reference system, where word forms are represented in their familiar orthography while word meanings are represented by synonym sets called synsets. Each synset consists of a list of synonymous words or collocations [6] and pointers that describe the relations between this synset.

  We used WordNet Version 3.0, which is the latest version available for download from the WordNet site [7].

- **Named Entity Overlap**: This feature relies on the use of Stanford Named Entity Recognizer [8] (Finkel et al., 2005) which is a Java implementation of a NER.

  This implementation offers well-designed feature extractors for NE recognition as well as several options for defining feature extractors. Good named entity recognizers for english are included with the download, which is composed by 7 classes: Organization,

[6] A collocation is a group of words that usually go together (e.g., 'take in', 'heavy rain')
[7] http://wordnet.princeton.edu/contact
[8] http://nlp.stanford.edu/software/CRF-NER.shtml

Person, Money, Time, Location, Percent, Date.

- **Edit distance**: This feature is based on the Levenshtein distance algorithm, where the source code to implement this distance is freely available [9].

4.3.3 Evaluation metrics

We emphasize that we are mainly based on the evaluation metrics used in CLEF for PR and selection tasks. The metrics used for the empirical evaluation of the performance of the passage extraction engine are the following:

- The accuracy @10: which is defined as the percentage of right answers compared to the total number of questions (in our case the accuracy is measured for the first 10 positions).

$$Accuracy = \frac{N_r}{N} \tag{4.18}$$

  where N denotes the number of questions and N_r denotes the number of questions correctly answered.

- The number of questions having correct passages ranked first.

- The Mean Reciprocal Rank (MRR) which denotes the multiplicative inverse of the rank position of the first correct answer, and it is defined as follows:

$$MRR = \frac{1}{N} \times \sum_{1=1}^{N} \frac{1}{R_i} \tag{4.19}$$

  where N represents the number of questions and R_i denotes the rank of correct answer to the question i.

In order to evaluate the performance of the whole approach, we were based on the following measures proposed by CLEF:

- The c@1 measure which was introduced as the major evaluation measure for both passage and answer selection tasks. The formula of $c@1$ is set to:

$$c@1 = \frac{1}{n}(n_R + n_U \frac{n_R}{n}) \tag{4.20}$$

  where n_R denotes the number of questions correctly answered, n_U represents the number of questions unanswered and n is the total number of questions. This measure is interpreted as follows:

[9] http://www.sanfoundry.com/java-program-implement-levenshtein-distance-computing-algorithm/

1. A system that returns an answer to all the questions receives a score equals to the accuracy measure since in this case $n_U = 0$, so, $c@1 = \frac{n_R}{n}$.

2. The unanswered questions add value to $c@1$ only if they do not reduce considerably the accuracy (i.e., n_R/n) achieved by the system for all the questions. That is to say, a system can achieve a great $c@1$ value if it can replace some *NoA* answers by the correct one.

3. A system that does not answer any question (i.e., gives only NOA answers) obtains a $c@1$ value equal to 0 as n_R=0 in both sides.

- #NoA: the number of questions unanswered.
- #R: the number of questions answered correctly.
- #W: the number of questions answered wrongly.
- #NoA R: the number of questions unanswered where a right candidate answer is discarded: In this case, the system chooses to leave the question unanswered (pessimistic behavior).
- #NoA W: the number of questions unanswered with wrong candidate answer.
- #NoA Empty: the number of questions unanswered with empty candidate: in which no candidate answer was given.
- Overall accuracy: the accuracy calculated over all assessed answers.

In Table 4.5, we summarize the main techniques reported by participants, namely nlel (Correa et al., 2010a), bpac (Nemeskey, 2010), dict (Sabnani & Majumder, 2010), elix (Agirre et al., 2010), iles (Agirre et al., 2010), ju_c (Pakray et al., 2010), uaic (Iftene et al., 2010), uiir (Toba et al., 2010b) and uned (Rodrigo et al., 2010). We remarked that by almost half of the systems that have reported the used retrieval model have employed Okapi BM25 [10], while other reported models have resorted to Lucene [11]. Note that some systems (ie., bpac, dict, elix, nlel, uaic, uiir, uned) have submitted two runs with a few differences in the implementation of the techniques used in both runs.

[10] Okapi BM25, also referred to as BM25, is a ranking function used by search systems in information retrieval to rank matching documents according to their relevance to a search query.

[11] Lucene is a free open source information retrieval software library, originally written in Java that works with text fields within document files.

Table 4.5 – Methods used by participating systems

System name	Retrieval Model	Linguistic Unit which is indexed				
		words	Lemmas	Stems	N-grams	Chunks/phrases
nlel	Distance Density, N-gram Model, BM25			x	x	
bpac	Okapi BM25	x		x		
dict				x		
elix	BM25			x		
iles			x			
ju_c	Apache Lucene	x	x	x	x	
uaic	Lucene	x				
uiir			x			
uned	BM25			x		

4.4 Results and discussion

As the passage extraction process greatly depends on the search step, we have tested two search models as indicated so far in order to select the best one and consider it in our n-gram model. The best model is the one that returns the correct candidates among the searched passages for most questions. The tested models were the tf-idf weighting mentioned in formula 4.2 and the model proposed by (Correa et al., 2010b) mentioned in formula 4.3. Out of a total of 198 questions, the latter returned 183 correct passages in the first 10 positions while the first one returned only 171 correct passages. Therefore, we apply the second search model in our system.

Recall that the idea behind our *Passim* formula is to keep the adjacent words of the question in the passage thanks to the factor (l_i/sng_i). The more grouped the words are, the higher the weight will be. Through our experiments, we found that the importance degree of this factor in the similarity calculation affects the results. Therefore, we used a parameter $\alpha \in [0.1]$ to set the level of importance of this factor. We considered $(l_i/sng_i)^{\alpha}$ instead of (l_i/sng_i) in the similarity formula. We tested different α values using the complete gold standard from CLEF 2010 as the development data. for the three languages, we obtain the best MRR values with α =0.1. For instance, we report in Figure 4.7 the results obtained in french, where the different curves for each of the evaluation criteria i.e, the MRR and the correct number of passages in the 10 first positions, show best values where α =0.1. So, we use this value in calculating the similarity measure *Passim*.

We now present the empirical results yielded by our approach compared to those obtained by NLEL System (Correa et al., 2010b) which includes a PR model based on n-grams. It was

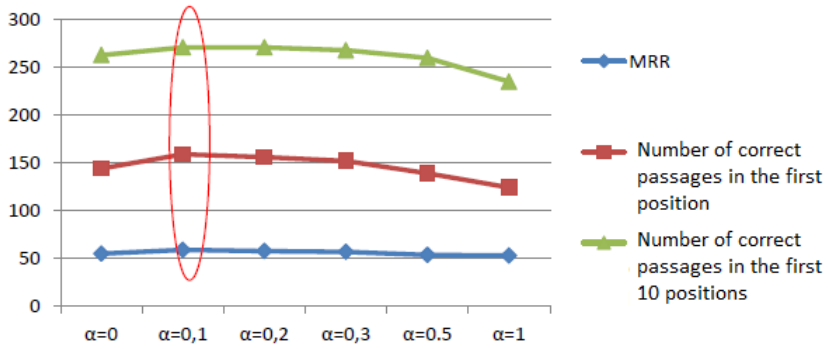


Figure 4.7 – The variation of the MRR and the number of correct passages according to the variation of the α value

ranked first in the CLEF PR track for French and Spanish and second for English language. The results presented in Table 4.6 show that our approach gives better results than NLEL for the 3 languages on all criteria. PaROD succeeds in answering a significant percentage of questions, with a difference equals 14% questions more than NLEL for English and 16% for both French and Spanish languages. We get more answers in the first position with a difference between 15 and 26 questions. For the remainder of the positions, for both systems, the number of questions is more important for the top positions. Besides, the MRR value obtained by PaROD is greater in the first positions and lower in the last ones. We can then deduce that our *Passim* measure is not only faster but also more efficient than that of NLEL.

Table 4.6 – Comparison between the PR module of PaROD and NLEL

Language	English		French		Spanish	
System	PaROD	NLEL	PaROD	NLEL	PaROD	NLEL
Percentage of questions having correct passages in the first 10 positions	84	78	80	76	81	77
Percentage of questions having a correct passage in the first position	51	44	47	42	48	44
Accuracy @10	0.849	0.784	0.804	0.760	0.811	0.754
MRR	0.451	0.385	0.409	0.365	0.410	0.371

Considering the overall approach, Table 4.7 presents the results yielded by our system run compared to the other previously described systems performing the same task of selecting the most relevant passage to answer a given question.

Table 4.7 – Comparison between PaROD and similar systems

System	Accuracy	c@1	#R	#W	#NoA	#NoA R	#NoA W
PaROD	0.76	0.85	152	23	25	0	0
uiir101PSenen	0.72	0.73	143	54	3	0	3
bpac102PSenen	0.68	0.68	136	64	0	0	0
dict102PSenen	0.67	0.68	117	52	31	17	14
bpac101PSenen	0.65	0.65	129	71	0	0	0
elix101PSenen	0.65	0.65	130	70	0	0	0
nlel101PSenen	0.64	0.65	128	68	4	2	2
uned102PSenen	0.65	0.65	129	71	0	0	0

In comparison to other related systems performing the same Paragraph selection task, PaROD shows superior performance in terms of accuracy and $c@1$ measures with a significant accuracy score equal to 0.74 and a higher $c@1$ score equal to 0.83. It is worth-noting that since the proposed $c@1$ value is greater than the accuracy score, the use of our no answer criterion was appropriate and has allowed to obtain a greater $c@1$ value. We mention that out of 99 complex questions, PaROD succeeds to answer 48 questions, where most of the unanswered and incorrectly answered questions were opinion or cause ones.

However, there is scope for further experiments on larger datasets to determine the threshold value for the ranking final score. It is worth noting that we have chosen not to deliver any candidate answer for unanswered questions, neither correct nor incorrect, thus, the number of unanswered questions is equal to the number of NoA Empty=25.

Although the paragraph selection task is just a PR, the major difference from pure IRs is to add the option of leaving the question unanswered in the validation step. Accordingly, this task allows posting complex questions and evaluating them in a simple way.

We emphasize that we have tested the overall approach using the english corpora as we have resorted to the english versions of major used tools deemed to achieve higher performance such as the english version of WordNet Lexical Database and the named entity recognizer for English. Obviously, we can also evaluate our approach in other languages merely by integrating multilingual tools.

4.5 Conclusion

In this Chapter, we have described our proposed approach for retrieving and ranking passages in the context of open domain QA and presented our experimental study based on CLEF datasets in

order to evaluate its performance. Our experimentation results have shown that our approach is competitive and multilingual giving motivating results compared to other state-of-the-art systems in different languages. We have proved that our proposed similarity measure based on n-gram structure named Passim is efficient as our passage extraction engine based on this latter has outperformed the system ranked first in CLEF PR exercise. We have gone beyond a simple extraction of passage list to deduce the most relevant one to the user's question. We have further demonstrated that integrating different similarity measures using RankSVM model, allows to better re-rank the retrieved passages and ensure the relevance of the passage delivered in response to a given natural language question. In the next Chapter, we will dig into a more challenging and crucial problem in cQA, namely question retrieval problem.

Chapter 5

Learning word embeddings for question retrieval in community QA

5.1 Introduction

Over the last years, with the boom of Web 2.0, the world has witnessed a large spread of user-generated content, which became a important information source on internet. This brings wide attention to the emerging concept of community Question Answering (cQA), which provides platforms for people with different backgrounds to share knowledge in the form of questions and answers. However, community services have rapidly built up huge archives of question-answer pairs that are continuously increasing accumulating duplicated questions. Therefore, users can hardly find the good answers and consequently post new queries that already exist in the archives.

In order to reduce the time lag required to get a new answer, it is critical to automatically search the community archive to check if equivalent questions have previously been posted. If a similar question is found, its associated answer can be directly returned as a relevant answer to the new query. Numerous studies have been recently done along this line as shown in Chapter 3, with the aim of answering new questions with past answers. Question retrieval (QR) is indeed a non trivial task facing several challenges as questions in cQA vary significantly in terms of vocabulary, length, structure and content quality. The major challenge is the word mismatch between the queried questions and the archived ones since users can phrase the same question using different wording. Word mismatch means that similar questions can be formulated such that they have different, but related words. For instance, the questions: *How can I slow down signs of aging naturally?* and *What are some home remedies to keep your skin looking younger?*

have almost the same meaning but include different words and then may regarded as dissim-

ilar. Most previous works on QR focus on enhancing the similarity measure between questions while it is difficult to set a compelling similarity function for discrete and sparse word representations. Recent efforts in word representations, have led to the rise of the emerging word embeddings, which have shown promise in various NLP tasks (G. Zhou et al., 2015; Musto et al., 2016; Esposito et al., 2020). Motivated by the tremendous success of these models, we propose a word embedding-based approach to retrieve similar questions in cQA.

This Chapter is structured as follows: In Section 5.2, we present our proposed word embedding based approach to improve the QR task and we detail its different components. In Section 5.3, we describe the experimental setup and discuss the obtained results in both English and Arabic. The final Section includes concluding remarks.

5.2 Description of the proposed WEKOS approach

The intuition behind our proposed approach for QR, called WEKOS (Word Embedding, Kmeans and COSine based approach), is to turn words in a community question into continuous low-dimensional vectors (Othman et al., 2019a). Unlike traditional methods which represent questions as Bag Of Words (BOWs), we suggest representing each question as a Bag of-Embedded-Words (BoEW) in a continuous space. The word embeddings are learned in advance using the continuous bag-of-words (CBOW) model (Mikolov, Chen, et al., 2013). The word vectors of a given question are weighted and averaged to get an overall representation of the question. We resort to the K-means clustering algorithm to create clusters from the collection of related questions. We believe that Kmeans provides a good strategy to reduce the data dimensionality and decrease the runtime cost of the search and ranking tasks. Therefore, each query is matched against the questions contained within its closest cluster rather than the entire collection of questions. The cosine similarity is employed to calculate the similarity between the average of the word vectors corresponding to the queried question and that of each existing question in the given cluster. The previous questions are thereafter ranked according to their cosine similarity scores in order to return the top ranking question as the most relevant one to the new posted query. As depicted in Figure 6.1, the WEKOS approach consists of five modules detailed below namely, question preprocessing, word embedding learning, embedding vector weighting, question clustering and question ranking.

5.2.1 Question preprocessing

Text preprocessing is a keyl task in NLP to evaluate and improve the quality of text data in order to ensure the validity and reliability of the statistical analysis. Our question preprocessing

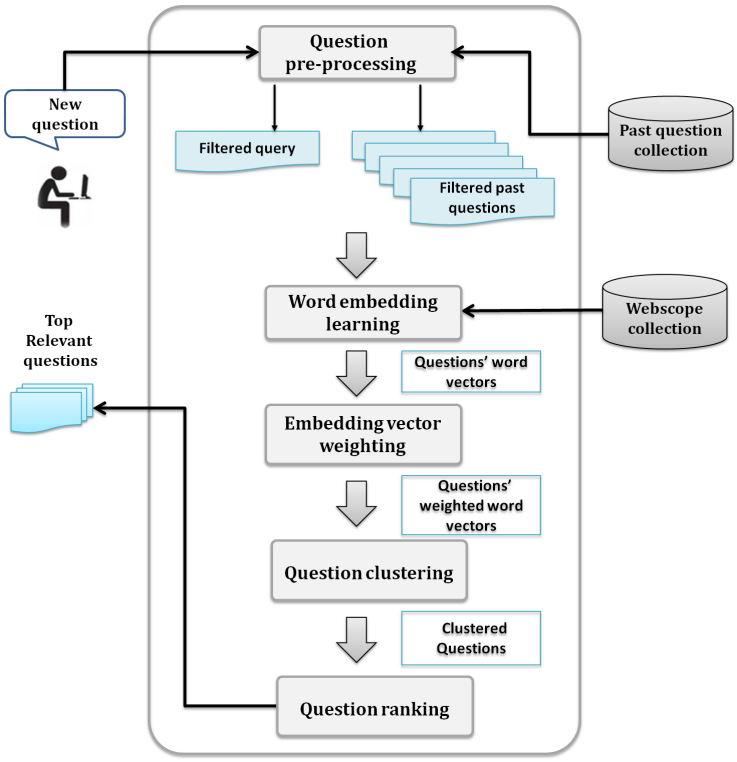


Figure 5.1 – WEKOS pipeline for question retrieval in cQA

module intends to filter the human natural language questions and extract the useful terms in order to represent them in a standard way. Basically, this module encompasses text cleaning, tokenization, stopwords removal and stemming. Punctuation marks, non letters, diacritics, and special characters such as &, #, $ and £are removed. English letters are lowercased while dates are normalized to the token *date* and numerical digits are normalized to the token *num*. At the end of the preprocessing module, we obtain a set of filtered queries, each of which is formally defined as follows:

$$q = \{t_1, t_2, ..., t_Q\} \tag{5.1}$$

where t represents a separate term of the query q and Q denotes the number of query terms. As for the Arabic question collection, aside from the aforementioned tasks, orthographic normalization is required to reduce noise and ambiguity in the Arabic text data. This task comprises Tachkil removal (ignoring arabic short vowels), Tatweel removal (deleting stretching symbol), and Letter normalization (variant forms to one form conversion). Indeed, certain spelling variants are sometimes inconsistently misued by Arabic writers, such as the Hamza; some may ignore it or employ a wrong Hamza variant. Whence, we normalize to one standard variant as follows: « أ ، إ ، آ ، ؤ ، ء ، ئ » are normalized to « ا ». For example, people always write المروءة instead

of المروءة. We then normalize it as follows: المرواة. In this way, words containing miswritten Hamzas will not be ignored.

5.2.2 Word embedding learning

This module is based on the use of word embeddings to build continuous word vectors based on their contexts in a huge corpus using shallow neural network. Word embeddings learn a low-dimensional vector for each vocabulary term in which the similarity between the word vectors can capture the syntactic and semantic similarities between the corresponding words. Basically, there exist two main types of word embeddings namely Continuous Bag-of-Words model (CBoW) and Skip-gram model. The former one consists in predicting a current word given its context, while the second does the inverse predicting the contextual words given a target word in a sliding window. It is worth noting that, in our approach, we consider the CBOW model (Mikolov, Chen, et al., 2013) to learn word embeddings, since it has proven through experiments to be more efficient and performs better with sizeable data than Skip-gram.

As depicted in Figure 5.6, the CBOW model predicts the center word given the representation of its surrounding words using continuous bag-of-words representation of the context, hence the name CBOW.

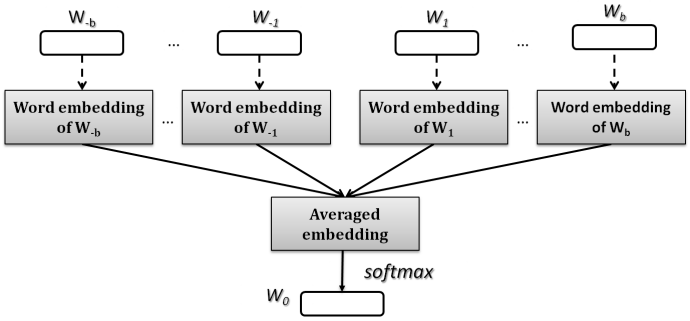


Figure 5.2 – Overview of the Continuous Bag-of-Words model.

The context vector is got by averaging the embeddings of each contextual word while the prediction of the center word w_0 is obtained by applying a softmax over the vocabulary V.

The goal of CBOW is to find the probability of a word occurring in a context. Let's consider a corpus with a sequence of words $\{w_1, w_2, ..., w_T\}$. The context vectors are summed and used to predict the target. Formally, CBOW aims to maximize the following objective function formula:

$$\frac{1}{T}\sum_{t=1}^{T} logP(w_t \mid \sum_{-c\leq j\leq c, j\neq 0} w_{t+j}) \tag{5.2}$$

Where T denoted the number of words in the corpus, w_t is a current word and c is the size of the context window. In practice, the size of the window is a random number which has a non-trivial effect on the resulting vector similarities.

5.2.3 Embedding vector weighting

The continuous word vectors of the questions are weighted using TF-IDF, which is one of the most commonly used term weighting schemes in IR systems owing to its simplicity and effectiveness. Each embedding word is multiplied by the TF-IDF of the word it represents. Recall that TF-IDF is a statistic weighting function that helps to estimate the importance of a word based on its relative frequency in a specific document and the inverse proportion of documents containing the word over the entire document collection. The TF-IDF weighting allows to have a suitable text representation for question comparison. As we work on questions, we adapt the basic TF-IDF function to our QA context by replacing documents with questions.

Given a question collection C, a word w and a question q, TF-IDF is defined as follows:

$$tfidf(w, q, C) = tf(w, q) * idf(w, Q) = f_{w,q} * \log(\frac{|C|}{df_{w,C}}) \tag{5.3}$$

where $f_{w,q}$ is the number of times w appears in a question q, $|C|$ is the size of the question collection and $df_{w,C}$ is the total number of questions that contain the word w.

TF-IDF was utilized to estimate the importance of a word not only in a particular question, but also in the full question collection. Indeed, some common words may appear several times in questions but they are not relevant as key-concepts to be indexed or searched. Intuitively, rare words that are common only in a single or few number of questions tend to have high scores while those which occur frequently in questions will be assigned low scores.

The weighted embedding vectors of the query words are averaged to obtain the average vector V_q of the queried question as follows:

$$V_q = \frac{\sum_{i=1}^{|V|}(\upsilon_{w_i} \times tfidf(w_i, q, C))}{\sum_{i=1}^{|V|} tfidf(w_i, q, C)} \tag{5.4}$$

where υ_{w_i} is the embedding vector of the word w_i generated by word2vec and $|V|$ is the number of word vectors in a given question q . Similarly, for each historical question, we compute its average vector V_d.

5.2.4 Question clustering

In order to achieve good performance, we propose to group similar questions together looking for the centers of each question cluster. We opt for the Kmeans (Hartigan & Wong, 1979) clustering algorithm, which is a popular unsupervised learning algorithm for data clustering, known for its simplicity and speed. The basic idea of the kmeans algorithm is to group items into k clusters of greatest possible distinction, where each item belongs to the cluster with the nearest mean. The clusters are represented by their centroïds. An item is considered to be in a particular cluster if it is closer to that cluster's centroïd than any other centroïd. Formally, K-means can be viewed as a heuristic algorithm to minimize the following objective function, known as squared error function:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \| x_i^{(j)} - c_j \|^2 \tag{5.5}$$

Where $\| x_i^{(j)} - c_j \|^2$ is a chosen distance measure between a data point and the cluster center c_j. k is the number of clusters and n is the number of cases.

The main algorithmic steps for k-means clustering are the following:

1. Randomly select K cluster centers (centroïds).
2. Assign each item to the group that has the closest centroïd according to the chosen distance function
3. When all objects have been assigned, recalculate the positions of the K centroïds.
4. Repeat steps 2 and 3 until convergence is achieved and the centroïds no longer move.

Although the kmeans process can always terminate, it does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm also significantly depends on the initial randomly selected cluster centers. Therefore, it can be run multiple times to reduce this effect.

In our approach, the distance calculation in K-means refers to the Euclidean distance. The one single parameter we need to set is K. Note that text clustering was performed at question level where the averaged word embeddings of the questions are feeded to kmeans.

5.2.5 Question ranking

The similarity between a query and a previous question in the vector space is calculated as the cosine similarity between their vectors q and d as follows:

$$CosSim(d, q) = \frac{q.d}{\| q \|_2 . \| d \|_2} \tag{5.6}$$

where q and d are the average of the word vectors of the queried question and the historical question, respectively. Each query is compared to the questions contained within its closest kmeans cluster instead of the entire community question collection. Questions are ranked according to their cosine similarity scores based on their weighted vectors in order to deliver the top ranking questions having the maximum score, as the most relevant ones to the new query. It is worthwhile to mention that the cosine similarity measure is a typical function that was widely used in previous work on word embeddings and has proven to be significantly effective in identifying closest words occurring in similar contexts and detecting their semantic similarity (Kenter & De Rijke, 2015; Levy et al., 2015).

5.3 Experiments

5.3.1 Datasets

Our experiments were conducted using the dataset released by (W.-N. Zhang et al., 2016) for QR evaluation. In order to construct the dataset, the authors harvested questions from all categories in the well-known Yahoo! Answers community platform, and then randomly splitted the questions into two subsets while maintaining their distributions in all categories. The former set represents the question repository for question search containing 1,123,034 questions, while the second set is the test set containing 252 original queries and 1624 manually annotated related questions. Annotators were hired to label the questions with "relevant" if a candidate question is considered semantically similar to the query or "irrelevant" otherwise. In case of conflict, a third annotator will make the decision for the final result. For example, in Table 5.1, Q_2 and Q_3 can be considered as *relevant* to Q_1 and their answers will then be used to answer the queried question Q_1, while Q_4 is labeled with *irrelevant* as it is not expected to satisfy Q_1 .

Table 5.1 – *An example of question retrieval*

Query
Q1- How to make a java chip?
Relevant
Q2- Can you help me to make my own Starbucks Java mint Starbucks Java mint Frappuccino?
Q3- Does anyone have a mint frappuccino recipe?
Irrelevant
Q4- How do you make a java web application?

Note that the questions in the test data do not overlap with those in the retrieval data. The number of similar questions related to each original query varies from 2 to 30.

The questions in the collection have different structures and belong to diverse categories including Business and Finance, Travel, Entertainment and Music, Computers and Internet, Sports, Beauty and Style, Pets, Health, Games and Recreation, Home and Garden, Society and Culture, etc. As shown in Histogram 5.3, the major categories including the largest proportions of questions are Travel, Pets, Health, Sports.

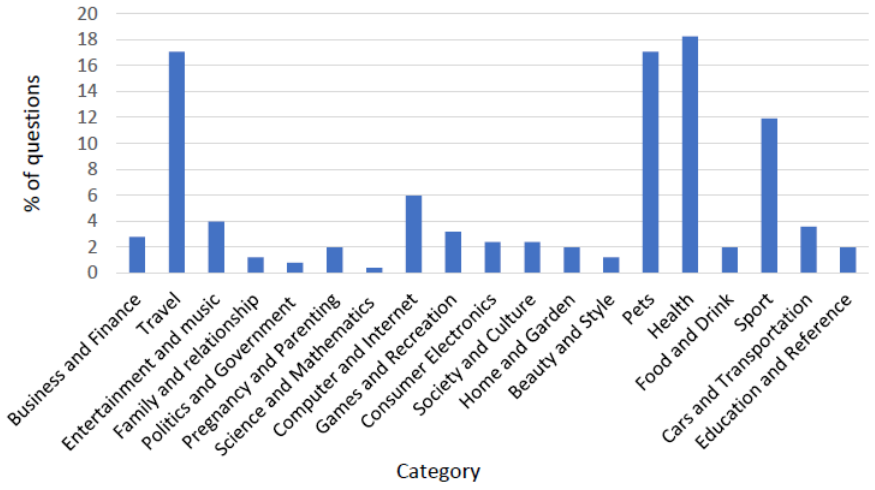


Figure 5.3 – Question distribution across different categories

The questions varies from 1 to 20 words as shown in the pie charts in Figures 5.4 and 5.5 for English and Arabic respectively.

Figure 5.4 – Distribution of questions' length for the English collection

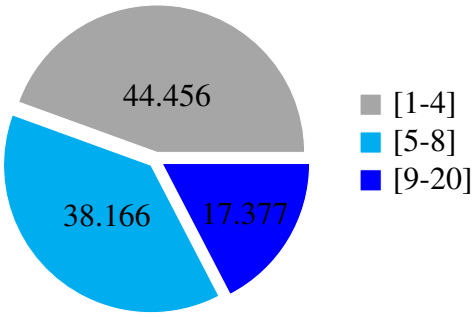
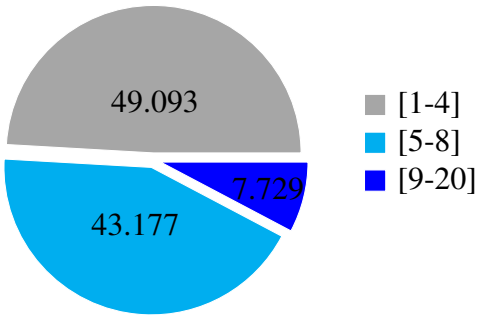


Figure 5.5 – Distribution of questions' length for the Arabic collection



We remark that Arabic questions are mostly shorter that the English ones mainly due to internal voweling which can denotes passive constructors. For example, the phrase *"He was dismissed"* can be translated to one single Arabic word «فصل». Tables 5.2 and 5.3 give examples of queries and their corresponding similar questions from the test sets in English and Arabic respectively. As can be seen in the tables 5.2 and 5.3, although the related questions are similar to the original queries asking for the same subject, they are worded differently.

Table 5.2 – *An example of community questions from the English test set.*

Query:	I often feel restless, uneasy, loss memory, lack of concentration, lose my temper. Why?
Category:	Health care
Topic:	Memory loss
Related questions	- I get short memory loss what should I do? - What to do when you are restless? - How can I improve my concentration and my memory or any mental exercise? - What is the best way to sharpen my brain?

Table 5.3 – *An example of community questions from the Arabic test set.*

Query:	كيف أقوم بتدريب كلب بلدي بالغ من العمر سنة واحدة ؟
Category:	Pets
Topic:	Puppy training
Related questions	-ما هي أفضل طريقة لتدريب جرو جديد؟ -هل لدى أحدهم اقتراح حول كيفية إيواء جرو عمره ١٠ أسابيع ؟ -كيف تدرب كلبًا يبلغ من العمر عاما ؟ -كيفية تدريب كلب صغير جداً ؟

To train the word embeddings, we used another sizeable data set extracted from cQA sites, namely the Yahoo! Webscope dataset[1], including 1,256,173 questions with 2,512,345 distinct words. As there is no large Arabic dataset available for the question retrieval task, for our experiments in Arabic we used the same English collection translated using Google Translation, the most-widely used free online machine translation tool. The Arabic data set contains the same number of questions as the English collection. The Arabic translated Yahoo! Webscope dataset, includes 2,512,034 distinct words, a bit fewer than the English set. Data preprocessing was performed before the experiments using NLTK [2]. After the preprocessing, the English corpus has been reduced by almost 15% and the Arabic one by nearly 20%. Note that the parameters of word2vec and Kmeans as well as kmeans were fixed using a parallel development set of 217 queries and 1317 annotated questions.

5.3.2 Evaluation metrics

In order to evaluate the performance of our proposed QR approach, we used Mean Average Precision (MAP) and Precision@n (P@n) as they are widely used for evaluating the QR task for cQA. In particular, MAP is the most commonly used metric in the literature estimating that the user is interested in getting several relevant questions for each original query. MAP rewards methods that not only return relevant results early, but also allow for a good ranking of the results. Given a set of queried questions Q, MAP denotes the mean of the average precision for each queried question q and it is set as follows:

$$\text{MAP} = \frac{\sum_{q \in Q} \text{AvgP}(q)}{\mid Q \mid} \tag{5.7}$$

[1]The Yahoo! Webscope dataset Yahoo answers comprehensive questions and answers version 1.0.2, available at "http://research.yahoo.com/Academic_Relations"

[2]https://www.nltk.org/

where $AvgP(q)$ is the mean of the precision scores after each relevant question q is

$$\text{AvgP} = \frac{\sum_r P@r}{R} \tag{5.8}$$

where r is the rank of each relevant question, R is the total number of relevant questions, and $P@r$ is the precision of the top-r retrieved questions.

Precision@n calculates the proportion of the top-n retrieved questions that are relevant to the given query. Given a set of queries Q, P@n is the proportion of the top n retrieved questions that are relevant to the queries and it is defined as follows:

$$\text{P@n} = \frac{1}{\mid Q \mid} \sum_{q \in Q} \frac{\text{Nr}}{N} \tag{5.9}$$

where Nr is the number of relevant questions among the top N ranked list returned for a query q. Note that in our experiments, we calculated P@10 and P@5. In order to fix the window size, we used the Accuracy which returns the proportion of correctly classified questions as relevant or irrelevant:

$$Accuracy = \frac{N_c}{Q} \tag{5.10}$$

with Q being the total number of queried questions and N_c the number of correctly classified questions.

Recall was also used for our evaluation, which returns the proportion of relevant similar questions that have been retrieved over the total number of relevant questions.

5.3.3 Word embedding learning and clustering

We trained our word embeddings on the whole Yahoo! Webscope dataset using word2vec in order to represent the words of the training data as continuous vectors which capture word-meaning and context. The training parameters of word2vec were set after numerous tests. For the English dataset, the word2vec parameters were set as follows:

- Size=300: feature vector dimension. We tested different values in the range $[50, 400]$ but did not get significantly different precision values. The best precision was achieved with size=300.
- Sample=1e-4: this is the down sampling ratio for the words that are very redundant in the corpus.

- Negative samples =25: the number of noise words
- min-count=1 : minimum number of words which we set to 1 to make sure we do not throw away anything.
- Context window=10: number of words considered around the pivot word. Considering that the window size is a crucial parameter for improving the accuracy of the retrieval method, we tested different window sizes and report the accuracy values obtained when querying the word embeddings generated with Skip-gram and CBOW models. Figure 5.6 shows that with our English corpus, CBOW outperforms Skip-grams in terms of accuracy and for both models, the optimal window size is 10. As a matter of fact, large windows tend to capture more context and topic information but increase the runtime needed to train the model while small windows tend to capture more information about the word itself, so picking out the optimal window size can improve the accuracy and reduce the computational time.

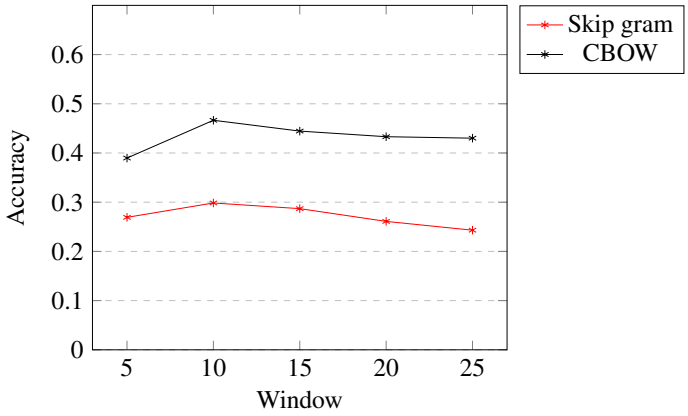


Figure 5.6 – Accuracy variations according to window size for CBOW and Skip-gram models with fixed dimensional vector model (size=300) for the English collection

For the Arabic dataset, the training configuration parameters of word2vec were set after many tests as follows:

- Size=300: We tested different values in the range $[50, 400]$ but did not get significant difference. The best precision was reached with size=300.
- Sample=1e-5
- Negative samples =15

- min-count=1 : We set the minimum number of words to 1 to not throw away anything.
- Context window=5: The window size is an important parameter affecting the resulting vectors, we tested different window sizes and report the accuracy values obtained when querying the word embeddings generated with both Skip-gram and CBOW models. Figure 5.6 shows that with our Arabic corpus, CBOW outperforms Skip-grams in terms of accuracy and for both models, the optimal window size is 5. So, we set the context window to 5 in order to reduce the runtime required to train word2vec.

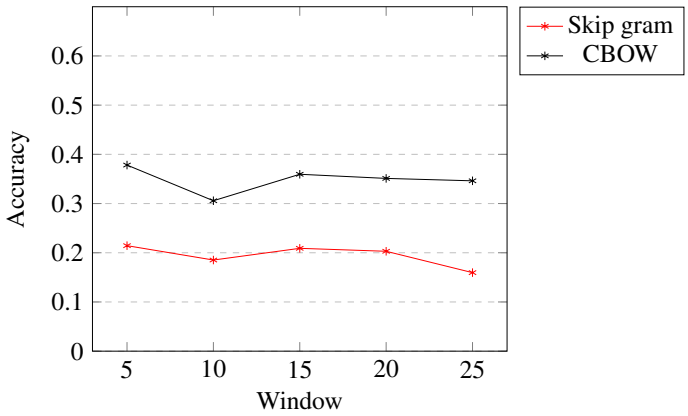


Figure 5.7 – Accuracy variations according to window size for CBOW and Skip-gram models with fixed dimensional vector model (size=300) for the Arabic collection

The number of clusters K is a crucial parameter to our approach, used for clustering the vocabulary of words. The clusters are used to obtain the question representations and matching. It is worth noting that fixing k is challenging as we need to make a trade-off between cluster quality, memory and runtime to create the clusters. After performing several experiments with this parameter by varying it in the range of [25, 200], we observed that the best results were reached with K equals 100 as as will be shown later in the Results Subsection (See Figures 5.8 and 5.9). Hence, we reported our results with this setting of the K parameter.

5.3.4 Results and discussion

We compare the performance of our approach WEKOS with different competitive state-of-the-art QR models tested by Zhang et al. in (W.-N. Zhang et al., 2016), on the same English dataset. The compared models are the following:

- **TLM** (Xue et al., 2008): A translation based language model which merges a translation-based language model with a query likelihood approach for the language model, for

the question and the answer parts respectively. TLM incorporates word-to-word translation probabilities learned by using different sources of information.

- **ETLM** (Singh, 2012): An entity based translation language model, which is an extension of TLM where the main difference is the substitution of the word translation by entity translation in order to integrate semantic information within the entities.
- **PBTM** (G. Zhou et al., 2011): A phrase based translation model which relies on machine translation probabilities, based on the assumption that question retrieval should be performed at the phrase level. PTLM learns the probability of translating a sequence of words in a previous question into another word sequence of words in a query.
- **WKM** (G. Zhou et al., 2013):A world knowledge based model which incorporates the Wikipedia knowledge into the questions by deriving the concept relationships that allow to detect related topics between the queries and the historical questions. WKM used Wikipedia as an external resource to add the estimation of the term weights to the ranking function. A concept thesaurus was created based on the semantic relations extracted from Wikipedia. The semantic relations are then leveraged to improve the question similarity in the concept space.
- **M-NET** (G. Zhou et al., 2015): A continuous word embedding based model, which integrates the category information of the questions to obtain a category based word embedding, estimating that the word representations belonging to the same category should be semantically equivalent.
- **ParaKCM** (W.-N. Zhang et al., 2016): A key concept paraphrasing based approach which explores the translations of pivot languages and expands questions with the paraphrases. It assumes that paraphrases can offer additional semantic connection between the key concepts in the queried question and those of the previous ones.

In Table 5.4, we summarize the main retrieval models on which the compared models are based.

Table 6.1 compares the performance of WEKOS with the aforedescribed models on the English Yahoo! Answers dataset. From Table 6.1 , we can see that PBTM outperforms TLM which proves that capturing contextual information in modeling the translation of phrases as a whole or consecutive sequence of words is more effective than translating single words independently. This is mainly because there is a dependency between adjacent words in a phrase.

ETLM, an extension of TLM, performs as good as PBTM which demonstrates that replacing the word translation with entity translation for question ranking can improve the

Table 5.4 – Overview of the compared models

Model name	Retrieval Model	Level based retrieval model
TLM	Translation based Language Model	Word
ETLM	Translation based Language Model	Entity
PBTM	Translation based Language Model	Phrase
WKM	World Knowledge based Model	Concept
M-NET	Continuous Word Embedding based Model	Word
ParaKCM	Key Concept Paraphrasing based Model	Concept

Table 5.5 – *Results comparing performance of WEKOS with other models on the English Yahoo! Answers dataset*

	TLM	**ETLM**	**PBTM**	**WKM**	**M-NET**	**ParaKCM**	**WEKOS**	**WEKOS without TF-IDF**
P@5	0.3238	0.3314	0.3318	0.3413	0.3686	0.3722	**0.4338**	**0.3431**
P@10	0.2548	0.2603	0.2603	0.2715	0.2848	0.2889	**0.3647**	**0.2736**
MAP	0.3957	0.4073	0.4095	0.4116	0.4507	0.4578	**0.5036**	**0.4125**

performance of the translation language model. Although, both ETLM and WKM rely on Wikipedia as an external knowledge resource, WKM employs wider information from the knowledge source. Particularly, WKM creates a Wikipedia thesaurus, which derives the concept relationships (e.g. synonymy, polysemy, hypernymy, and associative relations) based on the structural knowledge of Wikipedia. The different relations in the thesaurus are considered according to their importance to expand the queries and then enhance the traditional similarity measure for QR. However, the performance of WKM and ETLM are constrained by the low coverage of the Wikipedia concepts on the various community questions. M-NET, based on continuous word embeddings performs well owing to the integration of metadata of category information in the learning process to encode the properties of words, from which similar words can be grouped according to their categories. The best performance in terms of precision and was achieved by ParaKCM, a key concept paraphrasing based approach which explores the translations of pivot languages and expands queries with the generated paraphrases for question retrieval.

The results illustrate that our approach WEKOS outperforms in English all the compared methods on all criteria by returning a good number of relevant questions among the retrieved ones early. One possible reason is that context-vector representations learned by word2vec can effectively address the lexical gap problem by capturing semantic relations

between question words, while most of the other methods do not capture enough information about semantic equivalence. We can admit that the text representation given by bag-of-embedded words is more efficient and meaningful than traditional bag-of-words models which can detect neither semantics nor positions in text. This significant performance shows that the use of word embeddings along with TF-IDF weighting and cosine similarity is effective in the question retrieval task. Nonetheless, we found out that sometimes, our approach fails to retrieve similar questions: Out of 252 test questions, only 12 questions get P@10 values equal to zero. Most of these questions contain misspelled query terms. For example, questions containing the mistaken term *sofwar* cannot be retrieved for a query including the term *software*. Such a case reveals that our approach fails to address some lexical disagreement problems. Moreover, there are few cases where WEKOS is unable to detect semantic equivalence. These cases mostly include questions having one single similar question and most words of this latter do not appear in a similar context with those of the queried question, such as: *Which is better to aim my putter towards, the pole or the hole?* and *How do I aim for the target in golf?*. Obviously, further experiments with the dimensions of the embeddings are needed to improve the obtained results.

In addition, we tested our approach with and without TF-IDF weighting (In Table 6.1, WEKOS and WEKOS without TF-IDF respectively) to investigate its effect on the question retrieval results. Through our experiments, we can deduct that the use of TF-IDF allows to increase the P@5, P@10 and the MAP values. A possible reason behind this is that TF-IDF can detect questions that make frequent use of specific words and figure out if they are relevant in the question. We can say that the discriminatory power of TF-IDF enables the retrieval engine to capture relevant questions that could likely be similar to the new query. Nevertheless, in certain cases, a word can be relatively common over the whole collection but still holds some importance throughout the question like the words *date* and *system*. Such common words get a low TF-IDF score, and thus are pretty much ignored in the search process. Moreover, TF-IDF doesn't take into account synonymy relations between terms. For instance, if a user posted a question including the word *dwelling*, TF-IDF would not consider community questions that might be similar to this query but instead use the word *bungalow*. TF-IDF can not resolve the lexical ambiguity problem which is frequent in a community collection of informal and heterogeneous questions where the same concept may be expressed in various ways. Note that the computational complexity of TF-IDF is $O(nm)$, where n is the total number of words and m is the total number of questions in the corpus. For large collections like yours, this could present an escalating problem.

Table 6.4 illustrates the performance of WEKOS on the Arabic dataset.

As shown in Table 6.4, WEKOS had relatively good performance in Arabic, where the MAP has not exceeded 0.2916. We can say that the major reason for this is that the word embedding based model ignores the morphological structure of Arabic words. Indeed, the

Table 5.6 – *Question retrieval performance of WEKOS on the Arabic dataset*

	WEKOS	**WEKOS without TF-IDF**
P@5	**0.3444**	**0.2545**
P@10	**0.2412**	**0.1933**
MAP	**0.4144**	**0.2916**

nature of the Arabic language as an inflectional and a morphologically rich language [3] with high character variation has a significant impact on how influential a dataset is for delivering good word embeddings. Accordingly, exploiting the word internal structure is important to detect semantically similar words. For instance, the most similar words to «فعل» are all variants of the same word such as «فاعل ، سنفعل ، يفعلون ، فعلنا ، نفعل».

Consequently, enriching word embeddings with their main grammatical information (such as the word, person, number, gender, tense, case) could allow to produce more meaningful embeddings that capture morphological, context and semantic similarities. Interestingly, in terms of recall, we get 0.4677 and 0.3828 values for English and Arabic respectively, which implies that the number of omitted relevant questions is not big.

We fine-tuned the parameter k within 25 to 200 for the English and Arabic corpora. We observed that the more the k value increases, the more the clustering execution time increases, the more the search time decreases.

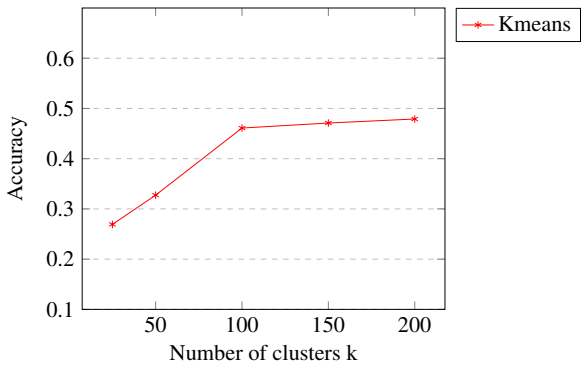


Figure 5.8 – Accuracy variations according to the number of clusters k for the English dataset

As shown in Figures 5.8 and 5.9, the accuracy reaches 0.4877 and 0.3780 for the English and Arabic datasets respectively, with k=100 and then continues to slightly hover over these values but does not much increase. Therefore, we set k to 100 as an estimated

[3]Morphologically Rich Languages (MRLs) refer to languages in which significant information concerning syntactic units and relations is expressed at word-level

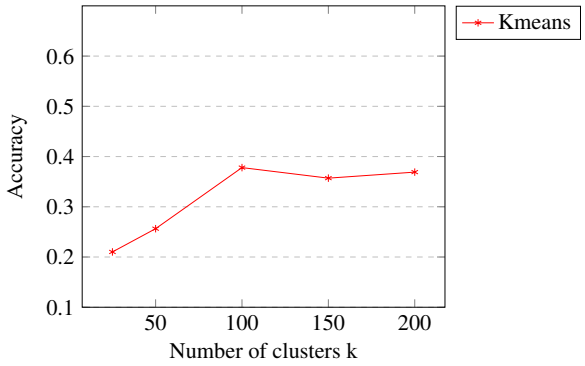


Figure 5.9 – Accuracy variations according to the number of clusters k for the Arabic dataset

value for both datasets to avoid increasing the clustering runtime. Obviously, further experiments are needed to determine the optimal k value that satisfies the trade-off between cluster quality and runtime. Despite being linear, fast and simple, the major drawback of the kmeans algorithm is its non-deterministic nature because it requires to pre-specify the number of clusters and it randomly selects the initial centroïds.

Thus, hierarchical clustering might be a good alternative clustering approach since it does not imply to pre-specify the number of clusters the way that k-means does.

5.4 Conclusion

In this Chapter, we addressed the challenging problem of question retrieval which is of significant importance to real-world cQA services. In order to solve the word mismatch between community questions, we proposed a word embedding based approach named WEKOS. Overall, the question words are embedded in a continuous space using word2vec and treated as a bag of embedded words. The produced word vectors are learned using the CBOW model and weighted based on the frequency of the words. We relied on the cosine similarity to calculate the similarity between the questions based on their vector-based word representations. K-means was applied on word embeddings to decrease the data dimension and improve the performance of the approach. Experiments conducted on large-scale cQA datasets in English and Arabic showed the effectiveness of the proposed approach in both languages in detecting similar questions even if they share few common words. The experimental results demonstrate that WEKOS achieves better results compared to other competitive methods. We have shown evidence that the TF-IDF weighting, though simple, can enhance the search efficiency as well as the quality of the retrieval results. In the next Chapter, we will dig further into the question retrieval problem trying to

improve it with a more effective approach relying on deep learning techniques.

Chapter 6

Attentive Siamese LSTM for question retrieval in cQA

6.1 Introduction

In this Chapter, we still focus on the problem of question retrieval (QR) in cQA which aims to retrieve from the community archives the previous questions that are semantically equivalent to new queries. Recall that the major challenges in this crucial task are the shortness of the questions as well as the word mismatch problem as users can formulate the same query using different wording. For instance, the questions *How can we relieve stress naturally?* and *What are some home remedies to help reduce feelings of anxiety?* like in Arabic, كيف يمكننا تخفيف التوتر بشكل طبيعي؟
and ماهي العلاجات المنزلية التي تساعد على تقليل الشعور بالقلق؟
have nearly the same meaning but include different words and therefore may be viewed as different.

While numerous attempts have been made to address this problem, most existing methods relied on the bag of-words (BOWs) representations which are constrained by their specificities that put aside the word order and ignore syntactic and semantic relationships. As shown in Chapter 3, recent successes in QR have been yielded using Neural Networks (NNs) (Dos Santos et al., 2015; Romeo et al., 2016; Mohtarami et al., 2016; Kamineni et al., 2018) which use a deep analysis of words and questions to take into consideration the semantics as well as the structure of questions in order to predict the semantic text similarity. Motivated by the tremendous success of these powerful models, in this Chapter, we propose an approach based on NNs in order to enhance the performance of our WEKOS approach and improve the QR task by more effectively detecting the semantic similarity

between the questions. We propose a new deep learning approach based on a Siamese architecture with LSTM networks, augmented with an attention mechanism. This latter is an additional layer, which lets the model give different words different attention while modeling questions. The semantic similarity between pairs of questions was predicted using a textual similarity measure. To evaluate the proposed approach, we conducted experiments on large-scale datasets in English and Arabic.

This Chapter is structured as follows: In Section 6.2, we present our proposed Siamese LSTM based approach to improve the QR task and we detail its different components. The experiments and the obtained results are outlined in Section 6.3. The final Section sums up the Chapter and contains concluding remarks.

6.2 Description of the proposed ASLSTM approach

In order to improve the QR task, we propose an Attentive Siamese LSTM approach for question retrieval, referred to as ASLSTM to retrieve the semantically similar questions in cQA (Othman et al., 2019b). As illustrated in Figure 6.1, our approach is composed of three main modules namely, question preprocessing, word embedding learning and Manhattan LSTM (MaLSTM).

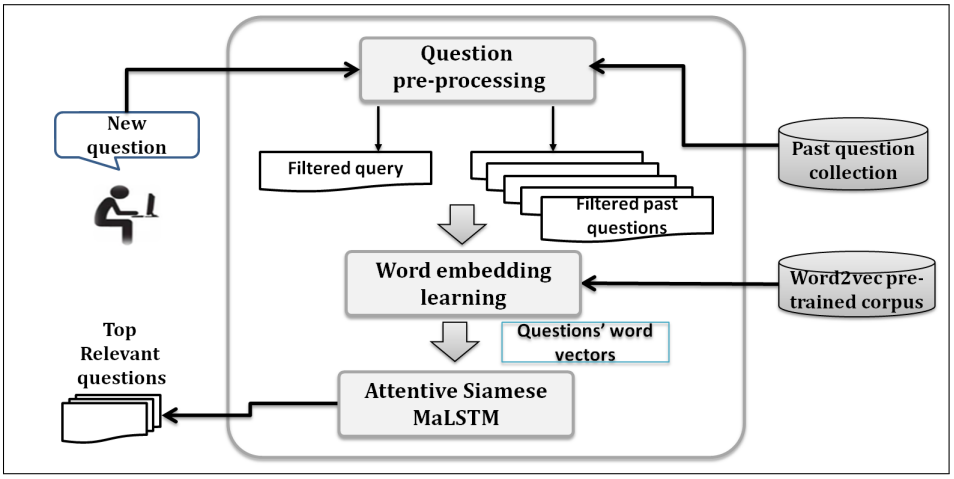


Figure 6.1 – ASLSTM pipeline for question retrieval in cQA

The basic principle underlying the ASLSTM approach is to map every question word token in to a fix-sized vector.

The word vectors of the questions are therefore fed to the Siamese LSTM with the aim of representing them in final hidden states encoding semantic meaning of the questions. An

attention mechanism is integrated in the Siamese architecture to determine which words should give more attention on than other words over the question. Community questions are then ranked by means of the Manhattan similarity function based on the vector representation of each question. A previous posted question is considered to be semantically equivalent to a queried question if their corresponding LSTM representations lie close to each other according to the Manhattan similarity measure. The historical question with the highest Manhattan score will be returned as the most similar question to the new posted one. The components of ASLSTM are detailed below.

6.2.1 Question preprocessing

Pre-processing is important to make the question collections cleaner and easier to process. Our question preprocessing module aims to filter the natural language community questions and extract the useful terms in order to represent them in a formal way. It is worth mentioning that the tasks of this module are exactly the same as those of the preprocessing module of the WEKOS approach described in Chapter 5. We remind that the preprocessing module comprises text cleaning, tokenization, stopwords removal and stemming. Punctuation marks, non letters, diacritics, and special characters are removed. English letters are lowercased while dates are normalized to the token *date* and numerical digits are normalized to the token *num*. For the Arabic question collection, in addition to the aforementioned tasks, orthographic normalization was applied, including Tachkil removal, Tatweel removal, and Letter normalization.

6.2.2 Word Embedding Learning

We remind that word embeddings are low-dimensional vector representations of words, learned by harnessing large amounts of text corpora using shallow neural networks. The use of word embeddings was relevant in our WEKOS approach and has allowed to effectively detect the syntactic and semantic similarities between words. Particularly, we resorted to the Continuous Bag-of-Words (CBOW) model which has proven to outperform Skip gram on our datasets. Recall that CBOW consists in estimating a pivot word according to its context using a window of contextual words around it, while Skip gram does the inverse predicting the contextual words given a current word in a sliding window. In our word embedding learning module, we map every word into a fix-sized vector using Word2Vec pretrained on an external corpus.

6.2.3 Attentive Siamese Manhattan LSTM

6.2.4 LSTM

Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997), is a powerful type of RNN widely used in deep learning, and has proven its capacity to capture long-term dependencies and model sequential data. Interestingly, LSTM helps prevent the vanishing gradient problem (Hochreiter, 1998) which is the main limitation of RNN. Gradient descent is an optimization algorithm that uses an iterative process to minimize a given function and improve the deep learning. The basic intuition is to adjust the weights of the model by determining the error function derivatives according to each member of the weight matrices in the model. To minimize the total loss, the gradient descent updates each weight in proportion to the derivative of the error with respect to that weight. LSTM is endowed with a memory cell that is capable of maintaining its state over time, and internal mechanisms called gates to regulate the information flow. The major reason for relying on LSTM in our approach is its proven performance in handling variable-length sequential data.

Given input vector x_t, hidden sate h_t and memory state c_t, the updates in LSTM are performed as follows:

$$i_t = sigmoid(W_i x_t + U_i h_{t-1} + b_i) \tag{6.1}$$

$$f_t = sigmoid(W_f x_t + U_f h_{t-1} + b_f) \tag{6.2}$$

$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{6.3}$$

$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \tag{6.4}$$

$$o_t = sigmoid(W_o x_t + U_0 h_{t-1} + b_0) \tag{6.5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6.6}$$

where i_t, f_t, o_t are input, forget, and output gates at time t, respectively. W_k, U_k are LSTM parameterized weight matrices, b_k represents the bias vector for each k in $\{i, f, c, o\}$ and $(\odot)$ denotes an element-wise product of matrices, known as the Hadamard product which is an entrywise multiplication.

6.2.5 Siamese Manhattan LSTM

The overall aim of our Siamese Manhattan LSTM model referred to as Siamese MaLSTM is to compare a pair of questions to decide whether or not they are semantically equivalent. We used the Siamese LSTM (Mueller & Thyagarajan, 2016) architecture which is known to have identical sub-networks LSTMleft and LSTMright that are passed vector representations of two sequences and return a hidden state encoding their semantic meaning.

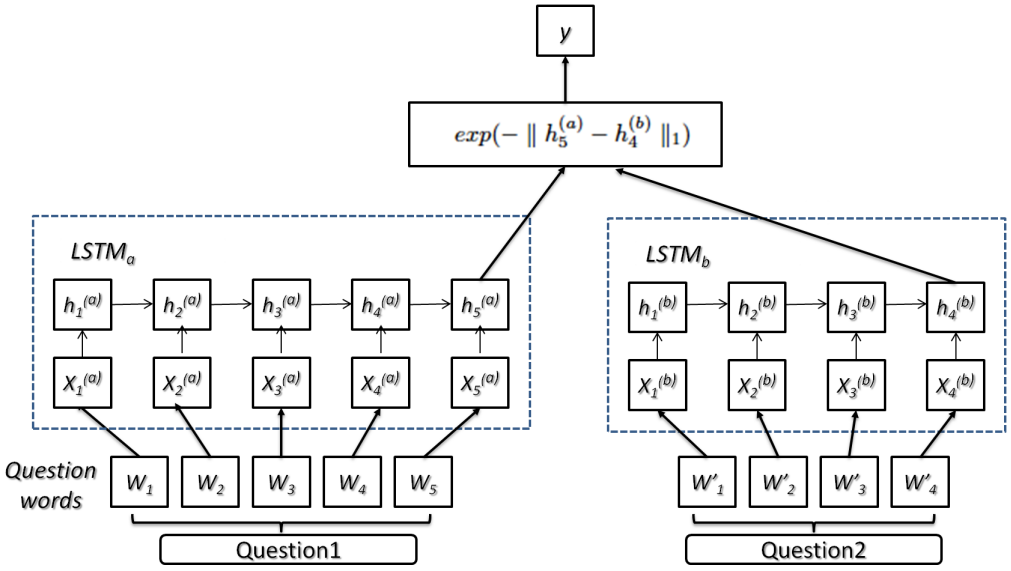


Figure 6.2 – General architecture of the Siamese MaLSTM model

Note that we decided to use LSTM for each sub-network, but it is also possible to swap LSTM with GRU (Gated Recurrent Unit). GRU is a variation on the LSTM, also aiming to solve the vanishing gradient problem which comes with a standard RNN. GRUs are almost similar to LSTMs in terms of design, although they have two gates, namely reset gate and update gate. Reset gate determines how to combine new input with previous memory while update gate is what input gate and forget gate were in LSTM, determining how much of the previous state to keep.

Although GRUs have less parameters and then might take less time to train, LSTMs empirically remember longer sequences than GRUs and usually outperform them in tasks requiring modeling long-distance relations. This is the reason why we opted for LSTM rather than GRU.

In our work, Siamese LSTM was adapted to the context of question retrieval, that is to say, the sentence pairs become pairs of questions.

LSTM learns a mapping from the space of variable length sequences d_{in} and encode the

input sequences into a fixed dimension hidden state representation d_{rep}. More concretely, each question represented as a word vector sequence (e.g., Q_1 is represented by x_1, x_2, x_3) is fed into the LSTM, which updates, at each sequence-index, its hidden state. The final state of LSTM for each question is a d_{rep}-dimensional vector, denoted by h in figure 6.2, which holds the inherent semantic meaning of the question.

Unlike vanilla RNN language models which predict next words, the given network rather computes the similarity between pairs of sequences. A major feature of the Siamese architecture is the shared weights across the sub-networks, which reduce not only the number of parameters but also the tendency of overfitting. Once we have the two vectors that capture the underlying meaning of each question, the semantic similarity between the questions is computed using the following Manhattan similarity function:

$$y = exp(- \parallel h^{(left)} - h^{(right)} \parallel_1) \tag{6.7}$$

Note that since we have an exponent of a negative, the Manhattan function scores will be between 0 and 1. It is worth mentioning that we tested different similarity metrics, namely Manhattan, cosine and Euclidean distances and the best results were obtained with the Manhattan distance as will be seen later in the next section.

6.2.6 Attention Mechanism

Attention mechanism with Neural networks have recently achieved tremendous success in several NLP tasks (P. Zhou et al., 2016; Chorowski et al., 2015; K. Xu et al., 2015). We assume that every word in a question contributes to the meaning of the whole question but the words do not have equal influential information. Thus, we should assign a probability to every word to determine how influential it is to the entire question. Note that we adopted an attention mechanism as in (Yang et al., 2016). Siamese LSTM model employs only the last hidden states of sequence pair e.g. $h_4^{(a)}$ and $h_5^{(b)}$, which may ignore some information. To remedy this problem, in our attention layer, we used all hidden states $H = \{h_1, h_2, ..., h_T\}$, where h_i is the hidden state of the LSTM at time step i summarizing all the information of the question up to x_i and T denotes the length of the question. The general architecture of the proposed Siamese Manhattan LSTM model augmented with an attention layer is illustrated in Figure 6.3, where the different constituent layers are shown from the input (question words) to the output (similarity score).

Note that $\alpha^{(a)}$ and $\alpha^{(b)}$ denote the weights of $LSTM_a$ and $LSTM_b$, respectively. Basically, the attention mechanism measures the importance of a word through a context vector u_h. It computes a weight α_i for each word annotation h_i according to its importance. The final question representation r is the weighted sum of all the word annotations using the

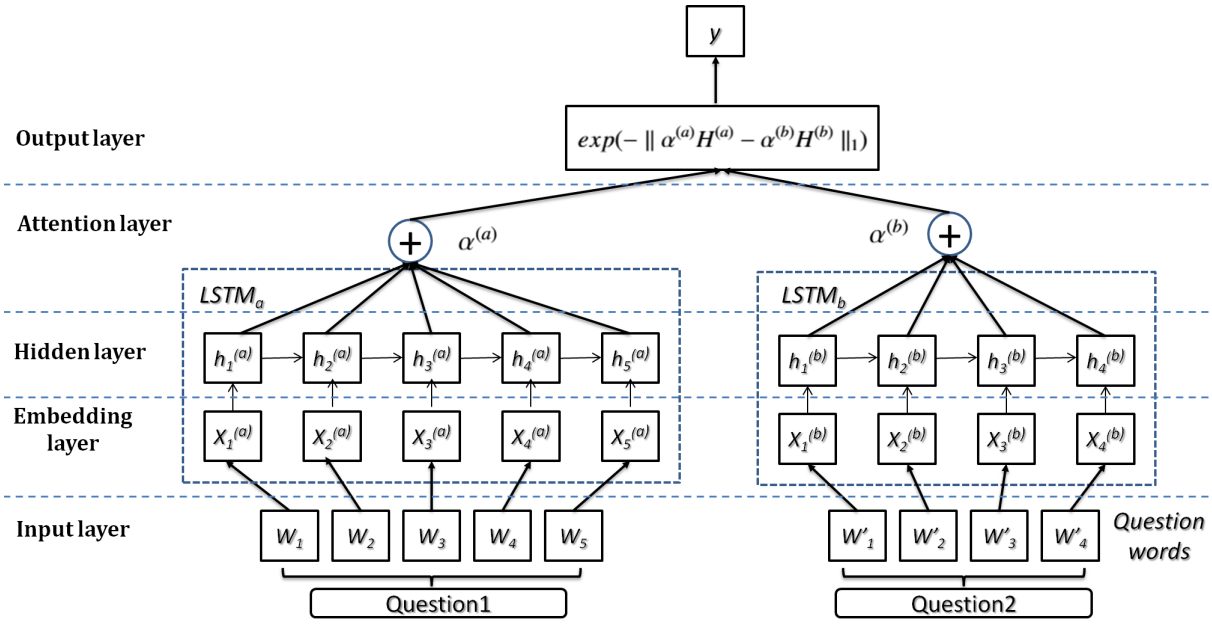


Figure 6.3 – General architecture of our Siamese Manhattan LSTM model with attention mechanism

attention weights, computed by equation 6.10. In the attention layer, a context vector u_h is introduced, which is randomly initialized and can be viewed as a fixed query, that allows to identify the informative words.

$$e_i = \tanh(W_h h_i + b_h), e_i \in [-1, 1] \tag{6.8}$$

$$\alpha_i = \frac{\exp(e_i^T u_h)}{\sum_{i=1}^{T} \exp(e_t^T u_h)}, \sum_{i=1}^{T} \alpha_i = 1 \tag{6.9}$$

$$r = \sum_{i=1} \alpha_i h_i, r \in R^{2L} \tag{6.10}$$

where W_h, b_h , and u_h are the learnable parameters with W_h is a weight matrix and b_h is a bias vector used to project each context vector into a common dimensional space.

6.3 Experiments

6.3.1 Datasets

In order to evaluate our propose approach, we performed experiments using the same dataset described in the previous Chapter released by (W.-N. Zhang et al., 2016) for evaluation. The questions of the community collection were harvested from all categories in

the popular Yahoo! Answers community platform, and then were randomly splitted into two sets while maintaining their distributions in all categories.

Recall that the first set is a question repository for question search containing 1,123,034 questions, while the second is the test set containing 252 queries and 1624 manually annotated related questions, where the number of relevant questions related to each original query varies from 2 to 30. The community questions in the collection are in various structures, different lengths and belonging to diverse categories e.g., Health, Sports, Computers and Internet, Diet and Fitness, Pets, Travel, Business and Finance, Entertainment and Music, Education and Reference, etc.

For our experiments in Arabic, we resorted to the same English collection translated using Google Translation, as there is no large Arabic dataset available for the question retrieval task. The Arabic collection includes exactly the same number of questions as the English set. In order to train word2vec for Arabic, we resorted to a sizeable data set from cQA sites, namely the Yahoo!Webscope dataset [1], translated into Arabic including 1,256,173 questions with 1 2,512,034 distinct words.

For Siamese LSTM training, we employed the publicly available Quora Question Pairs dataset [2]. The given collection encompasses 400,000 samples of question duplicate pairs, where each sample has a pair of questions along with ground truth about their corresponding similarity (1: similar, 0: dissimilar). A small excerpt of the train set is shown in 6.4, where in each row of the file, we have the unique identifier of the pair of questions "id", the first question id "qid1" and the second question id "qid2", the content of the first question "question1", the content of the second question " question2" and their corresponding similarity "is_duplicate" which can take values either 1 or 0; 1 if the pair are similar or 0 if they are not. A set of 40,000 pairs was employed for validation. The Quora test set was organized as pairs of questions as shown in Figure 6.5, where in each row, we find the unique identifier of the pair of questions "test_id", the content of the first question "question1" and the content of the second question "question2".

Following the same format, we organized our test set as pairs of questions to be directly fed into MaLSTM as shown in Figure 6.6, where each original question is associated with its similar questions as a set of pairs.

It is worth noting that data preprocessing was performed using Python NLTK.

[1]The Yahoo! Webscope dataset Yahoo answers comprehensive questions and answers version 1.0.2, available at "http://research.yahoo.com/Academic_Relations"

[2]www.kaggle.com/quora/question-pairs-dataset.

```
"id","qid1","qid2","question1","question2","is_duplicate"
"0","1","2","What is the step by step guide to invest in share market in india?","What is the step by step guide to invest in share marke
"1","3","4","What is the story of Kohinoor (Koh-i-Noor) Diamond?","What would happen if the Indian government stole the Kohinoor (Koh-i-N
"2","5","6","How can I increase the speed of my internet connection while using a VPN?","How can Internet speed be increased by hacking t
"3","7","8","Why am I mentally very lonely? How can I solve it?","Find the remainder when [math]23^{24}[/math] is divided by 24,23?","0"
"4","9","10","Which one dissolve in water quikly sugar, salt, methane and carbon di oxide?","Which fish would survive in salt water?","0"
"5","11","12","Astrology: I am a Capricorn Sun Cap moon and cap rising...what does that say about me?","I'm a triple Capricorn (Sun, Moon
"6","13","14","Should I buy tiago?","What keeps childern active and far from phone and video games?","0"
"7","15","16","How can I be a good geologist?","What should I do to be a great geologist?","1"
"8","17","18","When do you use シ instead of し?","When do you use ""&"" instead of ""and""?","0"
"9","19","20","Motorola (company): Can I hack my Charter Motorolla DCX3400?","How do I hack Motorola DCX3400 for free internet?","0"
"10","21","22","Method to find separation of slits using fresnel biprism?","What are some of the things technicians can tell about the du
"11","23","24","How do I read and find my YouTube comments?","How can I see all my Youtube comments?","1"
"12","25","26","What can make Physics easy to learn?","How can you make physics easy to learn?","1"
"13","27","28","What was your first sexual experience like?","What was your first sexual experience?","1"
"14","29","30","What are the laws to change your status from a student visa to a green card in the US, how do they compare to the immigra
"15","31","32","What would a Trump presidency mean for current international master's students on an F1 visa?","How will a Trump presiden
```

Figure 6.4 – A small excerpt of the train set

```
"test_id","question1","question2"
0,"How does the Surface Pro himself 4 compare with iPad Pro?","Why did Microsoft choose core m3 and not core i3 home Surface Pro 4?"
1,"Should I have a hair transplant at age 24? How much would it cost?","How much cost does hair transplant require?"
2,"What but is the best way to send money from China to the US?","What you send money to China?"
3,"Which food not emulsifiers?","What foods fibre?"
4,"How ""aberystwyth"" start reading?","How their can I start reading?"
5,"How are the two wheeler insurance from Bharti Axa insurance?","I admire I am considering of buying insurance from them"
6,"How can I reduce my belly fat through a diet?","How can I reduce my lower belly fat in one month?"
7,"By scrapping the 500 and 1000 rupee notes, how is RBI planning to fight against issue black money?","How will the recent move to
8,"What are the how best books of all time?","What are some of the military history books of all time?"
9,"After 12th years old boy and I had sex with a 12 years old girl, with her consent. Is there anything wrong?","Can a 14 old guy da
10,"What is the best slideshow app for Android?","What are the best app for android?"
11,"What services are from Google: Facebook, YouTube betray Twitter?","What social network (like Google, Facebook, WhatsApp, Viber, 
12,"What if a cricket hits a batsman's helmet and then goes to the boundary?","Should carbonated red balls and 8 yellow balls. If 5 
13,"Just how do you learn fruity loops?","How do Fruity Wrappers work?"
14,"Why does Batman get kill in Batman v Superman?","In Batman v Superman, why reduce Lex Luthor pit Superman against Batman?"
15,"When can I buy a SpaceX stock?","Should I sell or buy LNKD stock?"
16,"Is it gouging and price fixing?","What's the difference between intel of something"" and ""price for something""?"
17,"Can a vacuum cleaner concentrate suck your eye out if it is pressed against your face?","Could a vacuum cleaner suck get your ey
18,"I am 20 years old and I still a problem with pimples. What is the remedy?","I am 20 years old and still have acne. It seems more
19,"What is it ai living in the middle class?","Why middle class?"
20,"How matter at MIT? Will performing poorly in 11 grade affect my chance?","I have passed 5 AP tests with scores trump 5. Can I ap
21,"What possible with XAT percentile between 85 and 90?","Is it possible that a person getting below 90 percentile in weren CAT cra
22,"What are the differences between clients and servers?","What is the difference between a server and a database?"
```

Figure 6.5 – A small excerpt of the Quora test set

6.3.2 Word Embedding Learning

For English word embedding training, we resorted to the publicly available word2vec vectors [3], with dimensionality of 300, that were trained on 100 billion words from Google News. Since no Arabic version of Google News vectors yet exists, we train the translated Yahoo!Webscope dataset using the CBOW model, as it has proven through experimentation to be more efficient and performs better than Skip-gram with our data. The training parameters of the CBOW model on the Arabic collection were set after several tests as follows:

[3]https://code.google.com/p/word2vec/

```
"test_id","question1","question2"
0,"do you ride dressag do you know what dressag","doe anyon ride dressag jumper"
1,"do you ride dressag do you know what dressag","do you ride western english game pleasur dressag jump"
2,"do you ride dressag do you know what dressag","doe ani bodi know anyth about dressag"
3,"do you ride dressag do you know what dressag","what dressag"
4,"do you ride dressag do you know what dressag","what dressag train"
5,"do you ride dressag do you know what dressag","dressag horserid"
6,"how can i stop my cat worri nibbl her own leg","how can i stop my femal cat from knaw her leg from nervou disord"
7,"what ideal temperatur ball python","temperatur ball pyhthon incag"
8,"what ideal temperatur ball python","what best temperatur ball python' cage how often normal them shed"
9,"how do i get knot out my cat fur","how do you get knot clump out your cat fur"
10,"how do i get knot out my cat fur","how can i remov tangl my cat' fur"
11,"how do i make cage rat","how could i make rat cage doesnt take up alot room monei"
12,"how do i make cage rat","urgent how do you make c c cage"
13,"how do i make cage rat","how make guinea pig cage"
14,"how do i make cage rat","what do i need make c c cage possibl nb guinea pig"
15,"when trot hors bump me up forward still can't post after sever class any tip","help any tip learn post when trot"
```

Figure 6.6 – A small excerpt of our test set

- Size=300: feature vector dimension. Note that we tested different values in the range [50, 500] but did not get significant difference in terms of precision values. The best precision was reached with size=300.
- Sample=1e-4: down sampling ratio for the words that are very redundant in the corpus.
- Negative samples=25: number of noise words
- min-count=1: minimum number of words which we set to 1 to make sure we do not throw away anything.
- Context window=5: fixed window size.

6.3.3 LSTM Training

During LSTM training, we applied the Adadelta method (Zeiler, 2012) for weights optimization to automatically decrease the learning rate. Gradient clipping was also used with a threshold value of 1.25 to avoid the exploding gradient problem (Pascanu et al., 2013). Our LSTM layers' size is 50 and embedding layer's size is 300. We employed the back propagation and small batches of size equals 64, to reduce the cross-entropy loss and we resorted to the Mean Square Error (MSE) as a common regression loss function for prediction. We trained our model for several epochs to observe how the results varied with the epochs. We found out that the accuracy changed with the variation of the number of epochs but stabilized after epoch 25. The given parameters were set based on several empirical tests; each parameter was tuned separately on a development set to pick out the best one. For implementing

our model we used Keras[4] and Scikit-learn[5]. Note that we used the same LSTM configuration for both languages (English and Arabic).

6.3.4 Evaluation Metrics

To evaluate our approach, we used Mean Average Precision (MAP), Precision@n (P@n) and Recall as they are the most widely used metrics for assessing the performance of question retrieval in cQA. We remind that MAP assumes that the user is interested in finding many relevant questions for each query and then rewards methods that not only return relevant questions early, but also get good ranking of the results. Precision@n gives an idea about the classifier's ability of not labeling a positive sample as a negative one. It returns the proportion of the top-n retrieved questions that are equivalent. Recall is the measure by which we check how well the model is in finding all the positive samples of the dataset. It returns the proportion of relevant similar questions that have been retrieved over the total number of relevant questions. We also used Accuracy, which returns the proportion of correctly classified questions as relevant or irrelevant.

6.3.5 Results and Discussion

In order to test the performance of ASLSTM, we compare it against our previous approach called WEKOS as well as the competitive state-of-the-art question retrieval methods tested by Zhang et al. in (W.-N. Zhang et al., 2016) on the same dataset. The methods being compared are recalled below:

* **WEKOS** (Othman et al., 2019a): A word embedding based method which transforms words in each question into continuous vectors. The questions are clustered using Kmeans and the similarity between them was measured using cosine similarity based on their weighted continuous valued vectors.
* **TLM** (Xue et al., 2008): A translation based language model which uses a translation-based language model with a query likelihood approach for the question and the answer parts respectively. TLM integrates word-to-word translation probabilities learned by using different sources of information.
* **ETLM** (Singh, 2012): An entity based translation language model, which is an extension of TLM where the major difference is the replacement of the word

[4]https://keras.io/
[5]https://scikit-learn.org

translation with entity translation in order to integrate semantic information within the entities.

* **PBTM** (G. Zhou et al., 2011): A phrase based translation model which uses machine translation probabilities assuming that QR should be performed at the phrase level. PTLM learns the probability of translating a sequence of words in a historical question into another word sequence of words in a given query.

* **WKM** (G. Zhou et al., 2013): A world knowledge based model which integrates the knowledge of Wikipedia into the questions by deriving the concept relationships that allow to identify related topics between the queries and the previous questions. A concept thesaurus was built based on the semantic relations extracted from Wikipedia.

* **M-NET** (G. Zhou et al., 2015): A continuous word embedding based model, which integrates the category information of the questions to get a category based word embedding, supposing that the representations of words belonging to the same category should be semantically equivalent.

* **ParaKCM** (W.-N. Zhang et al., 2016): A key concept paraphrasing based approach which explores the translations of pivot languages and expands queries with the paraphrases. It assumes that paraphrases give additional semantic connection between the key concepts in the queried question and those of the historical ones.

Table 6.1 gives a comparison of the performance of ASLSTM against the aforementioned models on the English Yahoo! Answers dataset. The results in Table 6.1, show that PBTM outperforms TLM which demonstrates that detecting contextual information in modeling the translation of entire phrases or consecutive word sequences is more effective than translating separate words, as there is a dependency between adjacent words in a phrase.

Table 6.1 – *Question retrieval performance comparison of different models in English.*

	TLM	**ETLM**	**PBTM**	**WKM**	**M-NET**	**ParaKCM**	**WEKOS**	**ASLSTM**
P@5	0.3238	0.3314	0.3318	0.3413	0.3686	0.3722	0.4338	**0.5033**
P@10	0.2548	0.2603	0.2603	0.2715	0.2848	0.2889	0.3647	**0.4198**
MAP	0.3957	0.4073	0.4095	0.4116	0.4507	0.4578	0.5036	**0.5799**

ETLM performs as well as PBTM, which proves that entity translation is more efficient than the word translation for ranking and could enhance the performance of the

translation language model. WKM is based on Wikipedia as an external knowledge resource to derive the concept relationships, but its performance is limited by the low coverage of the Wikipedia concepts on the diverse users' questions. ParaKCM achieves good precision by exploring the translations of pivot languages and expanding queries with the produced paraphrases for question retrieval. M-NET, also based on word embeddings, performs well owing to the use of metadata of category information to derive the properties of words. WEKOS based on word embedding too along with TF-IDF weighting and kmeans, achieves comparable results and further proves that the use of word embeddings get benefits from dense word representation and mitigate the negative impact of word mismatch by capturing semantic relations between words, while the other methods mostly do not capture enough information about the semantic similarity.

As illustrated in Table 6.1, our proposed approach ASLSTM outperforms in English all the compared methods on all criteria by successfully returning a significant number of similar questions among the retrieved ones. This good performance indicates that the use of Siamese LSTM along with attention mechanism Manhattan similarity is effective in the QR task. Word embeddings allow to obtain an efficient input representation for LSTM, capturing syntactic and semantic information in a word level. Interestingly, our approach does not require an extensive feature generation owing to the use of a pre-trained model. The results show that our Siamese based approach performs better than translation and knowledge based methods, which provides evidence that the question representations made by the Siamese LSTM subnetworks can learn the semantic relatedness between pairs of questions and then are more adequate for representing questions in the question similarity task. The Siamese network was trained using backpropagation-through-time under the MSE loss function which compels the LSTM sub-networks to detect textual semantic difference during training. A key virtue of LSTM is that it can accept variable length sequences and map them into fixed length vector representations which can overcome the length and structure's problems in cQA.

Another significant finding is the effectiveness of the attention mechanism which was able to improve the performance of the approach. We assume that the attention mechanism managed to boost the similarity learning process by assigning a weight to each element of the question. This weights will then allow to compute which element in the sequence the neural network should more attend.

Wekos averages the weighted embeddings, which is one of the most simple and widely used techniques to derive sequence embedding but it leads to losing the word order, while in our approach, the LSTMs update their state to get the main context meaning of the text sequence in the order of words. The goal of the Siamese archi-

tecture is to learn a function which can map a question to an appropriate fixed length vector which is favor for similarity measurement. Interestingly, it offers vector representation for a very short text fragment that should grasp most of the semantic information in that fragment.

In order to properly assess the MaLSTM model performance on our similarity prediction problem, we plot training data vs validation data accuracy and loss using the Matplotlib library. The training history of the model is often used to diagnose its behavior and to check whether it is a good fit for the data or could perform better with a different configuration. The loss is computed on training and validation and it shows how well the model is doing for these two sets. It denotes the sum of the errors made for each instance in training or validation sets. Loss is often used to determine the best parameter values for the given model. Obviously, the lower the loss, the better a model is. Once we find the optimized parameters, the accuracy is used to evaluate how accurate our model's prediction is compared to the true data. The training accuracy shows how much the model learns to map the input and output, while the validation accuracy tells about its generalizing ability. For instance, a decreasing validation accuracy means low generalization over the training data.

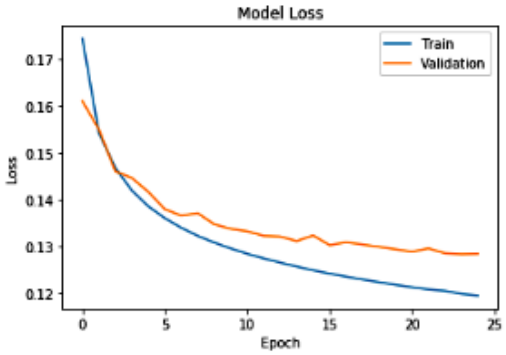


Figure 6.7 – Epochs vs loss of MaLSTM on the English dataset

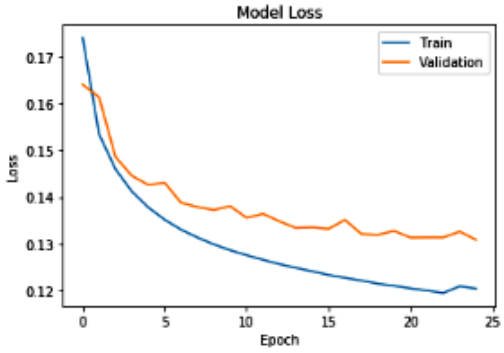


Figure 6.8 – Epochs vs loss of MaLSTM on the Arabic dataset

From Figures 6.7 and 6.8 which depict the training and validation set loss against the number of epochs [6], we can see that for both English and Arabic there is no considerable difference between the training and validation loss. The training loss keeps decreasing after every epoch which means that the model is learning to recognize the specific patterns. Similarly, the validation loss continues to decrease reaching 0.132

[6]Epoch is when the full dataset is passed both forward and backward through the neural network only once. It usually contains a few iterations

and 0.129 for English and Arabic respectively thus, our model is generalizing well on the validation sets. We can say that we have a good fit since both the train and validation loss decreased and leveled off around the same points.

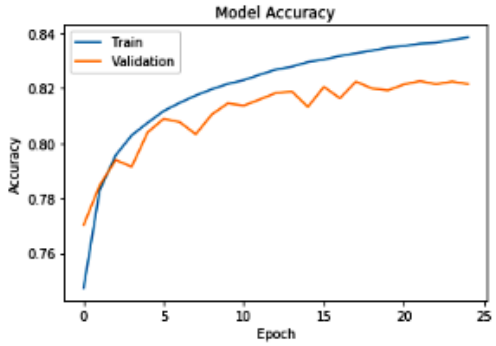


Figure 6.9 – Epochs vs Accuracy of MaLSTM on the English dataset

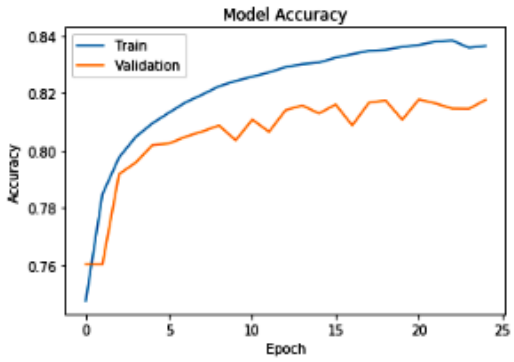


Figure 6.10 – Epochs vs Accuracy of MaLSTM on the Arabic dataset

From the plots of accuracy given in Figures 6.9 and 6.10, we observe that we get about 82% and 81% accuracy rate on the validation data for English and Arabic respectively. The model has comparable consistent accuracy on both train and validation sets. Both training and validation accuracy continue to increase without a sudden decrease of the validation accuracy, indicating a good fit. Therefore, we can admit that, whilst the performance on the training set is slightly better than that of the validation set in terms of both loss and accuracy, the model has converged to a stable value without any typical overfitting signs such as the continuous improvement of the training performance, while validation performance worsens.

It is worth mentioning that the accuracy used in the epochs-accuracy plots, is the binary accuracy calculated by Keras, and it implies that the threshold is set at 0.5 so, everything above 0.5 will be considered as correct.

We utilized the Manhattan distance which forces the LSTM model to entirely detect the semantic differences during training. In practice, our results are fairly stable across different similarity functions, namely cosine and Euclidean distances. We found that the Manhattan distance has outperformed them on both the English and Arabic datasets as depicted in Tables 6.2 and 6.3 which demonstrates that it is the most relevant measure for the case of high dimensional text data.

The cosine distance has outperformed the Euclidean distance which proves that it is better at catching the semantic of the questions, considering that the direction of the text points can be thought as its meaning, texts with similar meanings will have

Table 6.2 – Comparison between similarity measures on the English dataset

	P@5	**Recall**
Manhattan	**0.5033**	**0.5477**
Cosine	**0.3893**	**0.4345**
Euclidean	**0.3393**	**0.3843**

Table 6.3 – Comparison between similarity measures on the Arabic dataset

	P@5	**Recall**
Manhattan	**0.3702**	**0.4146**
Cosine	**0.2562**	**0.3006**
Euclidean	**0.2062**	**0.2506**

similar cosine score. Another reason is that Cosine distance is calculated using the dot product and magnitude of each vector. Thus, it is only affected by the words the two vectors have in common, whereas the Euclidean measure has a term for every dimension which is non-zero in either vector. We can therefore say that the Cosine distance has meaningful semantics for ranking texts, based on mutual term frequency, whereas Euclidean distance does not.

Moreover, we remarked that ASLSTM could find the context mapping between certain expressions mostly used in the same context such as *bug* and *error message* or also *need help* and *suggestions*. In addition, ASLSTM was able to retrieve similar questions containing certain common misspelled terms like *recieve* instead of *receive*, but it failed to capture other less common spelling mistakes like *relyable* or *realible* instead of *reliable*. Such cases show that our approach can address some lexical disagreement problems. Furthermore, there are few cases where ASLSTM fails to detect semantic equivalence, including queries having only one similar question and most words of this latter do not appear in a similar context with those of the query .

Table 6.4 – *Question retrieval performance of ASLSTM in Arabic*

	WEKOS	**ASLSTM**
P@5	0.3444	**0.3702**
P@10	0.2412	**0.2872**
MAP	0.4144	**0.4540**
Recall	0.3828	**0.4146**

Table 6.4 shows that our approach outperforms in Arabic the best compared system which gives evidence that it can also perform well with complex languages.

Nevertheless, a major limitation of our approach is that it ignores the morphological structure of Arabic words. As a matter of fact, the Arabic language is a morphologically-rich and complex language which expresses multiple levels of information at the word level. The variation in character forms appearing in handwritten Arabic has a notable impact on the generation of word embeddings. There-

fore, harnessing the word internal structure is crucial to capture semantically similar words.

Accordingly, endowing word embeddings with grammatical information such as, the person, gender, number and tense could help to obtain more meaningful embeddings that detect morphological and semantic similarity. In terms of recall, ASLSTM reaches 0.4136 for Arabic which implies that the number of omitted similar questions is not big. Interestingly, unlike traditional RNNs, Siamese LSTM is able effectively handle the long questions and learn long range dependencies thanks to its use of memory cell units that can store information across long input sequences. Nevertheless, for very long sequences, LSTM might fail to compress all information into its representation. Thus, the integration of the attention mechanism was relevant to let the model attend to all past outputs and give different words different attention while modeling questions.

6.4 Conclusion

In this Chapter, we have presented an attentive Siamese LSTM based approach, aiming at solving the question retrieval problem, which is of great importance in real-world cQA. For this purpose, we suggested using Siamese LSTM to capture the semantic similarity between the community questions. Our significant innovation was to add an attention mechanism to let the model give different attention to different words while modeling questions.

Siamese neural network architecture and attention mechanisms are notable recent trends in deep learning and have proven to achieve outstanding results in several tasks such as semantic textual similarity and machine translation. Our ASLSTM approach was among the few attempts to integrate Siamese architecture and attention mechanism in the question retrieval problem. For instance, (Homma et al., 2016) used GRU network instead of LSTM in their Siamese architecture to encode each question, and tested different distance measures to predict equivalence based on the question vector outputs of the neural network. A Siamese architecture without attention mechanism was used by Kamineni et al. (2018), who combined LSTMs and CNNs where the LSTM layer was used to learn representations for a given pair of questions and the CNN layer was employed for matching these representations and predict if they are semantically similar. Romeo et al. (2017) integrated an attention mechanism in the LSTM network as a selection model in their tree- kernel-based ranker to identify the most important text pieces in questions and then filter out noisy subtrees from their syntactic trees.

Experiments on large scale Yahoo! Answers datasets showed that our ASLSTM proposed approach can successfully improve the question similarity task in English and Arabic and outperform some competitive methods evaluated on the same dataset. Interestingly, we showed that Siamese LSTM is capable of modeling complex semantics and covering the context information of question pairs. However, a major limitation was the ignorance of the morphological structure of Arabic words. Furthermore, while it could address some lexical disagreement problems, our approach wasn't able to retrieve similar questions containing non-common misspelled terms.

Chapter 7

Conclusion

7.1 Contributions and limitations

In this thesis, we have presented our contributions in the field of Question Answering (QA), which has been subject of tremendous interest over the years as it has become a major asset to the user and a real challenge in many application areas such as e-commerce, distance learning and mobile search.

We have addressed two critical matching problems in open domain QA and community QA (cQA), namely passage retrieval (PR) and question retrieval (QR). PR is deemed to be the key component of a typical QA system, aiming to reduce the search space from a huge collection of documents to a fixed number of passages. Importantly, the performance of this component highly impacts that of the entire system. Most existing approaches tackling this problem were developed for closed domain with limited capacities, and they are no more than simple adaptations of classical document retrieval engines which are not especially devoted to QA and consequently cannot ensure high precision.

In order to improve the PR task, we proposed a novel PR and ranking approach for open domain QA named PaROD. Our contributions include a new n-gram based method for PR based on the degree of closeness or dispersion of question n-gram words in the passage. The extracted passages are re-ranked using a Ranking SVM model that combines a set of text similarity measures, including our proposed n-gram similarity measure as well as other lexical, syntactic and semantic features which have shown promise in the Semantic Textual Similarity task (STS) of SemEval. Our results have shown that the proposed approach PaROD could outperform in English other state-of-the-art systems using the CLEF data collection. We have

proved that our new n-gram based similarity measure is efficient since our passage extraction engine has outperformed in different languages the system ranked first in CLEF PR exercise. Additionally, we have demonstrated that integrating different similarity measures using RankSVM model allows to better re-rank the retrieved passages and ensure the relevance of the passages returned in response to a given natural language question. However, PaROD fails to answer some questions, where most of the unanswered and incorrectly answered questions were opinion or cause ones. Furthermore, the overall approach was only tested on the english corpora as we have resorted to the english versions of major used tools such as the english version of WordNet Lexical Database and the named entity recognizer for English. Further tests on different languages are required to verify that the PaROD approach is language independent. Also, we can say that the performance of PaROD highly depends on that of the external resources such as the lexical database and the named entity recognizer. Besides, our work lacks a detailed analysis of the approach complexity which allows to have an idea about the running time and space consumption of the program.

On the other hand, question retrieval in cQA while being similar to passage retrieval in QA, remains more challenging mainly due to the lexical gap problem. Most existing works on QR focus on the similarity measure between questions while it is tricky to set a compelling similarity function for sparse and discrete word representations. Recent efforts in word representations, have led to the rise of word embeddings, which have shown success in various NLP tasks. As we believe that word representations are crucial for the question retrieval task and motivated by the success of these models, we proposed a word embedding-based approach to retrieve similar questions in cQA named WEKOS. We suggested to turn words in a community question into continuous vectors using the CBOW model and weight the word vectors using TF-IDF. The K-means clustering algorithm was integrated to create clusters from the question collection of related questions. We believe that Kmeans provides a good strategy to reduce the data dimensionality and decrease the runtime cost of the search and ranking tasks. Therefore, each query is matched against the questions contained within its closest cluster rather than the entire question collection. The cosine similarity was used to calculate the similarity between the questions and rank them accordingly.

Our experiments conducted on a recently released large-scale cQA datasets showed the effectiveness of the proposed approach in English and Arabic in terms of detecting similar questions even if they share few common words. The results demonstrate that WEKOS can outperform other competitive methods by capturing semantic relations between question words, while most of the other methods do not detect enough

information about semantic equivalence. We have also shown evidence that the TF-IDF weighting can improve the search efficiency as well as the quality of the retrieval results. However, TF-IDF doesn't take into account synonymy relations between terms and can not resolve the lexical ambiguity problem which is frequent in a community collection of informal and heterogeneous questions where the same concept may be expressed in various ways. Also, the computational complexity of TF-IDF could be an escalating problem for large data collections. Besides, despite being simple, linear and fast, the main shortcoming of the kmeans clustering algorithm is its non-deterministic nature since it requires to pre-specify the number of clusters and it randomly selects the initial centroïds.

Although WEKOS has yielded good results, it could yet be improved by overcoming the limitations of TF-IDF and Kmeans and put more focus on the Arabic language which is as a morphologically rich language with high character variation and has a significant impact on delivering good word embeddings.

Our second proposed approach named ASLSTM has pushed work on question retrieval to the next level relying on the recent trends in deep learning namely, Siamese neural network architecture and attention mechanisms which have proven to achieve great results in various NLP tasks. In order to enhance the performance of our WEKOS approach and improve the QR task, we proposed a new deep learning approach based on a Siamese architecture with LSTM networks, augmented with an attention mechanism, which is an additional layer, integrated to let the model give different words different attention while modeling questions. The semantic similarity between questions was predicted using the Manhattan distance. Our experiments on large scale Yahoo! Answers datasets showed that our ASLSTM proposed approach can successfully improve the question retrieval task in both English and Arabic and outperform WEKOS and other competitive methods evaluated on the same dataset. We proved that Siamese LSTM is capable of modeling complex semantics and covering the context information of question pairs. Nevertheless, a major limitation was the ignorance of the morphological structure of Arabic words. Moreover, while it could address some lexical disagreement problems, our approach failed to retrieve similar questions containing non-common misspelled terms.

7.2 Future directions

Whilst promising approaches to passage retrieval in QA and question retrieval in cQA have been presented in this thesis allowing our research questions to be answered, there are several research directions we can follow to further improve our

work. We summarize the main routes for future research:

* **Experimenting with syntactic features in passage and question retrieval**

  In the processes of retrieving relevant passages and similar questions, we mainly relied on lexical, semantic and context information to measure the text similarity. There is still much room to experiment with more features such as syntactic ones to facilitate the semantic similarity measure between passages or questions. We believe that syntactic information is important for the ranking task and could improve the retrieval performance across morphologically rich languages such Arabic. Regarding PR, this could be done by integrating, along with the different features fed into RankSVM, a syntactic similarity measure that calculates the similarity between the syntactic trees of the passages produced by a syntactic parser. For QR, it would be interesting to incorporate morphological features into the embedding model or using LSTM to identify the best subtrees in the syntactic parsing of the questions, which will be then used in the ranking process.

* **Incorporating metadata in the learning process**

  CQA websites usually produce different types of metadata for the posted questions and answers such as category, voting score, tags, favourite count, last edit date, user expertise information and so on. Using this additional information may help to improve the performance of our QR approaches. Therefore, we are planning to take advantage of this information in our future work by incorporating various types of metadata information in the learning process in order to enrich the word representations.

* **Experimenting with different languages and datasets**

  Our proposed approaches in this thesis were tested on the CLEF and Yahoo!Answer datasets mostly in English and Arabic. In the future, we plan to continue our experiments with more diverse languages and larger datasets and carrying out an error and complexity analysis to compare the relative merits and demerits of our approaches.

* **Going conversational** A QA system should be equipped with at least a short term memory to remember the questions that were asked before and it should be able to track changes in structure and topic. This is a core element in building conversational QA systems. An exciting area of future work could be investigating the potential of attentive LSTM models with representation learning for the task of next question prediction in conversations which is analogue to the question retrieval problem in cQA.

References

Abacha, A. B., & Zweigenbaum, P. (2015). MEANS: A medical question-answering system combining NLP techniques and semantic web technologies. *Information Processing and Management*, *51*(5), 570–594.

Agirre, E., Ansa, O., Arregi, X., De Lacalle, M. L., Otegi, A., & Saralegi, X. (2010). Document expansion for cross-lingual passage retrieval. In *CLEF (notebook papers/LABs/workshops)*.

Araki, J., & Callan, J. (2014). An annotation similarity model in passage ranking for historical fact validation. In *Proceedings of the 37th international ACM SIGIR conference on research and development in information retrieval* (pp. 1111–1114).

Bae, K., & Ko, Y. (2019). Improving question retrieval in community question answering service using dependency relations and question classification. *Journal of the Association for Information Science and Technology*, *70*(11), 1194–1209.

Barrón-Cedeno, A., Da San Martino, G., Romeo, S., & Moschitti, A. (2016). Selecting sentences versus selecting tree constituents for automatic question ranking. In *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers* (pp. 2515–2525).

Bengio, Y., Simard, P., Frasconi, P., et al. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157–166.

Bilotti, M. W., Elsas, J., Carbonell, J., & Nyberg, E. (2010). Rank learning for factoid question answering with linguistic and semantic constraints. In *Proceedings of the 19th ACM international conference on information and knowledge management* (pp. 459–468).

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Blooma, M. J., Kurian, J. C., et al. (2011). Research issues in community based question answering. In *Proceedings of the 15th pacific asia conference on information systems* (p. 29).

Bobrow, D. G., Kaplan, R. M., Kay, M., Norman, D. A., Thompson, H., & Winograd, T. (1977). Gus, a frame-driven dialog system. *Artificial intelligence*, *8*(2), 155–173.

Buscaldi, D., Le Roux, J., Flores, J. J. G., & Popescu, A. (2013). Lipn-core: Semantic text similarity using n-grams, wordnet, syntactic analysis, esa and information retrieval based features. In *Proceedings of the 2nd joint conference on lexical and computational semantics* (p. 63).

Buscaldi, D., Rosso, P., Gómez-Soriano, J. M., & Sanchis, E. (2010). Answering questions with an n-gram based passage retrieval engine. *Journal of Intelligent Information Systems*, *34*(2), 113–134.

Buscaldi, D., Tournier, R., Aussenac-Gilles, N., & Mothe, J. (2012). Irit: Textual similarity combining conceptual similarity with an n-gram comparison method. In *Proceedings of the first joint conference on lexical and computational semantics-volume 1: Proceedings of the main conference and the shared task, and volume 2: Proceedings of the 6th international workshop on semantic evaluation* (pp. 552–556).

Cai, L., Zhou, G., Liu, K., & Zhao, J. (2011). Learning the latent topics for question retrieval in community qa. In *Proceedings of 5th international joint conference on natural language processing* (pp. 273–281).

Callan, J. P. (1994). Passage-level evidence in document retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 302–310).

Cao, X., Cong, G., Cui, B., & Jensen, C. S. (2010). A generalized framework of exploring category information for question retrieval in community question answer archives. In *Proceedings of the 19th international conference on World Wide Web* (pp. 201–210).

Cao, X., Cong, G., Cui, B., Jensen, C. S., & Zhang, C. (2009). The use of categorization information in language models for question retrieval. In *Proceedings of the 18th acm conference on information and knowledge management* (pp. 265–274).

Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., & Hon, H.-W. (2006). Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 186–193).

Chali, Y., Hasan, S. A., & Mojahid, M. (2015). A reinforcement learning formulation to the complex question answering problem. *Information Processing and Management*, *51*(3), 252–272.

Chen, L., Jose, J. M., Yu, H., Yuan, F., & Zhang, D. (2016). A semantic graph based topic model for question retrieval in community question answering. In *Proceedings of the ninth acm international conference on web search and data mining* (pp. 287–296).

Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-based models for speech recognition. In *Advances in neural information processing systems* (pp. 577–585).

Clarke, C. L., Cormack, G. V., Kisman, D. I., & Lynam, T. R. (2000). Question answering by passage selection (multitext experiments for TREC-9). In *TREC*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*(Aug), 2493–2537.

Correa, S., Buscaldi, D., & Rosso, P. (2010a). Nlel at RespubliQA 2010. In *CLEF (notebook papers/LABs/workshops)*.

Correa, S., Buscaldi, D., & Rosso, P. (2010b). NLEL-MAAT at RespubliQA. In *Multilingual information access evaluation i. text retrieval experiments* (pp. 223–228). Springer.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Cui, H., Sun, R., Li, K., Kan, M.-Y., & Chua, T.-S. (2005). Question answering passage retrieval using dependency relations. In *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 400–407).

da Silva, J. W. F., Venceslau, A. D. P., Sales, J. E., Maia, J. G. R., Pinheiro, V. C. M., & Vidal, V. M. P. (2020). A short survey on end-to-end simple question answering systems. *Artificial Intelligence Review*, 1–25.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, *41*(6), 391.

Dos Santos, C., Barbosa, L., Bogdanova, D., & Zadrozny, B. (2015). Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 2: Short papers)* (Vol. 2, pp. 694–699).

Duan, H., Cao, Y., Lin, C.-Y., & Yu, Y. (2008). Searching questions by identifying question topic and question focus. In *Acl* (Vol. 8, pp. 156–164).

Dudognon, D., Hubert, G., & Ralalason, B. J. V. (2010). Proxigénéa: Une mesure de similarité conceptuelle. In *Proceedings of the colloque veille stratégique scientifique et technologique (VSST 2010)*.

Echihabi, A., & Marcu, D. (2003). A noisy-channel approach to question answering. In *Proceedings of the 41st annual meeting on association for computational linguistics-volume 1* (pp. 16–23).

Esposito, M., Damiano, E., Minutolo, A., De Pietro, G., & Fujita, H. (2020). Hybrid query expansion using lexical resources and word embeddings for sentence retrieval in question answering. *Information Sciences*, *514*, 88–105.

Fader, A., Zettlemoyer, L., & Etzioni, O. (2014). Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*

(pp. 1156–1165).

Faiz, R., & Othman, N. (2019). Retrieving relevant passages using n-grams for open-domain question answering. *International Journal on Artificial Intelligence Tools*.

Fellbaum, C. (1998). *Wordnet: An electronic lexical database*. MIT.

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., et al. (2010). Building Watson: An overview of the deepQA project. *AI magazine*, *31*(3), 59–79.

Ferrucci, D., Nyberg, E., Allan, J., Barker, K., Brown, E., et al. (2009). Towards the open advancement of question answering systems. *IBM, Armonk, NY, IBM Res. Rep*.

Finkel, J. R., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics* (pp. 363–370).

Gaizauskas, R., & Wilks, Y. (1998). Information extraction: Beyond document retrieval. *Journal of Documentation*, *54*(1), 70–105.

Gao, J., Nie, J.-Y., Wu, G., & Cao, G. (2004). Dependence language model for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 170–177).

Gómez, J. M., Buscaldi, D., Rosso, P., & Sanchis, E. (2007). JIRS language-independent passage retrieval system: A comparative study. In *Proceedings of the 5th international conference on natural language processing (icon-2007)* (pp. 4–6).

Green Jr, B. F., Wolf, A. K., Chomsky, C., & Laughery, K. (1961). Baseball: an automatic question-answerer. In *Papers presented at the may 9-11, 1961, western joint IRE-AIEE-ACM computer conference* (pp. 219–224).

Hartigan, J. A., & Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100–108.

Hartrumpf, S. (2005). Question answering using sentence parsing and semantic network matching. In *Multilingual information access for text, speech and images* (pp. 512–521). Springer.

Haykin, S. (1994). *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

Herbrich, R., Graepel, T., & Obermayer, K. (1999). Support vector learning for ordinal regression. In *Proceedings of the 9th international conference on artificial neural networks* (pp. 97–102).

Hiemstra, D. (2009). Information retrieval models. *Information Retrieval: searching*

in the 21st Century, 2–19.

Hirschman, L., & Gaizauskas, R. (2001). Natural language question answering: the view from here. *Natural Language Engineering*, *7*(04), 275–300.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, *6*(02), 107–116.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Homma, Y., Sy, S., & Yeh, C. (2016). Detecting duplicate questions with deep learning. In *Proceedings of the 30th conference on neural information processing systems (nips 2016). ieee* (pp. 1–8).

Hovy, E. H., Hermjakob, U., & Lin, C.-Y. (2001). The use of external knowledge of factoid qa. In *TREC* (Vol. 2001, pp. 644–52).

Huang, E. H., Socher, R., Manning, C. D., & Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th annual meeting of the association for computational linguistics: Long papers-volume 1* (pp. 873–882).

Iftene, A., Trandabat, D., Husarciuc, M., & Moruz, M. A. (2010). Question answering on romanian, english and french languages. In *CLEF (notebook papers/LABs/workshops)*.

Ittycheriah, A., Franz, M., Zhu, W.-J., Ratnaparkhi, A., & Mammone, R. J. (2000). IBM's statistical question answering system. In *TREC*.

Izacard, G., & Grave, E. (2020). Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Jeon, J., Croft, W. B., & Lee, J. H. (2005). Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM international conference on information and knowledge management* (pp. 84–90).

Ji, Z., Xu, F., Wang, B., & He, B. (2012). Question-answer topic model for question retrieval in community question answering. In *Proceedings of the 21st ACM international conference on information and knowledge management* (pp. 2471–2474).

Joachims, T. (2002). *Learning to classify text using support vector machines: Methods, theory and algorithms*. Kluwer Academic Publishers.

Kamineni, A., Shrivastava, M., Yenala, H., & Chinnakotla, M. (2018). Siamese lstm with convolutional similarity for similar question retrieval. In *2018 international joint symposium on artificial intelligence and natural language processing (isai-nlp)* (pp. 1–7).

Katz, B., & Lin, J. (2003). Selectively using relations to improve precision in question answering. In *Proceedings of the workshop on natural language*

processing for question answering (EACL 2003) (pp. 43–50).

Keikha, M., Park, J. H., Croft, W. B., & Sanderson, M. (2014). Retrieving passages and finding answers. In *Proceedings of the 2014 australasian document computing symposium* (p. 81).

Kenter, T., & De Rijke, M. (2015). Short text similarity with word embeddings. In *Proceedings of the 24th acm international on conference on information and knowledge management* (pp. 1411–1420).

Krikon, E., Carmel, D., & Kurland, O. (2012). Predicting the performance of passage retrieval for question answering. In *Proceedings of the 21st acm international conference on information and knowledge management* (pp. 2451–2454).

Ku, H., Francis, W. N., et al. (1967). Computational analysis of present-day american english.

Laurent, D., Séguéla, P., & Nègre, S. (2005). Qristal, système de questions-réponses. In *TALN and RECITAL, tome 1 - conférences principales* (pp. 53–62).

Lee, G. G., Seo, J., Lee, S., Jung, H., Cho, B.-H., et al. (2001). SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. In *TREC*.

Lee, J., Seo, M., Hajishirzi, H., & Kang, J. (2020). Contextualized sparse representations for real-time open-domain question answering. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 912–919).

Levy, O., Goldberg, Y., & Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, *3*, 211–225.

Light, M., Mann, G. S., Riloff, E., & Breck, E. (2001). Analyses for elucidating current question answering technology. *Natural Language Engineering*, *7*(04), 325–342.

Lin, Y., Ji, H., Liu, Z., & Sun, M. (2018). Denoising distantly supervised open-domain question answering. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 1736–1745).

Liu, Y., Bian, J., & Agichtein, E. (2008). Predicting information seeker satisfaction in community question answering. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval* (pp. 483–490).

Llopis, F., & Vicedo, J. (2002). IR-n: A passage retrieval system at CLEF-2001. In *Evaluation of cross-language information retrieval systems* (pp. 244–252).

Lohr, S. (2012). The age of big data. *New York Times*, *11*.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1* (pp. 142–150).

Majumder, P., Mitra, M., & Chaudhuri, B. (2002). N-gram: a language independent approach to ir and nlp. In *International conference on universal knowledge and language.*

Malhas, R., Torki, M., & Elsayed, T. (2016). Qu-ir at semeval 2016 task 3: Learning to rank on arabic community question answering forums with word embedding. In *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)* (pp. 866–871).

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval* (Vol. 1). Cambridge university press Cambridge.

Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, *19*(2), 313–330.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Mohtarami, M., Belinkov, Y., Hsu, W.-N., Zhang, Y., Lei, T., Bar, K., … Glass, J. (2016). Sls at semeval-2016 task 3: Neural-based approaches for ranking in community question answering. In *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)* (pp. 828–835).

Moriceau, V., Tannier, X., & Grau, B. (2009). Utilisation de la syntaxe pour valider les réponses à des questions par plusieurs documents. In *CORIA* (pp. 5–18).

Moschitti, A., & Quarteroni, S. (2011). Linguistic kernels for answer re-ranking in question answering systems. *Information Processing and Management*, *47*(6), 825–842.

Mueller, J., & Thyagarajan, A. (2016). Siamese recurrent architectures for learning sentence similarity. In *Thirtieth aaai conference on artificial intelligence.*

Musto, C., Semeraro, G., de Gemmis, M., & Lops, P. (2016). Learning word embeddings from wikipedia for content-based recommender systems. In *European conference on information retrieval* (pp. 729–734).

Naili, M., Chaibi, A. H., & Ghezala, H. H. B. (2017). Comparative study of word embedding methods in topic segmentation. *Procedia computer science*, *112*, 340–349.

Nakov, P., Hoogeveen, D., Màrquez, L., Moschitti, A., Mubarak, H., Baldwin, T.,

& Verspoor, K. (2017). Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th international workshop on semantic evaluation (semeval-2017)* (pp. 27–48).

Nemeskey, D. M. (2010). SZTAKI@ RespubliQA 2010. In *CLEF (notebook papers/LABs/workshops).*

Ofoghi, B., & Yearwood, J. (2009). Can shallow semantic class information help answer passage retrieval? In *AI 2009: Advances in artificial intelligence* (pp. 587–596). Springer.

Ofoghi, B., Yearwood, J., & Ghosh, R. (2006). A semantic approach to boost passage retrieval effectiveness for question answering. In *Proceedings of the 29th australasian computer science conference-volume 48* (pp. 95–101).

Othman, N., & Faiz, R. (2016a). A multi-lingual approach to improve passage retrieval for automatic question answering. In *International conference on applications of natural language to information systems* (pp. 127–139).

Othman, N., & Faiz, R. (2016b). Question answering passage retrieval and reranking using n-grams and svm. *Computación y Sistemas*, *20*(3), 483–494.

Othman, N., Faiz, R., & Smaïli, K. (2019a). Enhancing question retrieval in community question answering using word embeddings. In *Proceedings of the 23rd international conference on knowledge-based and intelligent information engineering systems (KES).*

Othman, N., Faiz, R., & Smaïli, K. (2019b). Manhattan siamese lstm for question retrieval incommunity question answering. In *OTM confederated international conferences "on the move to meaningful internet systems"* (Vol. 11877).

Pakray, P., Bhaskar, P., Pal, S., Das, D., Bandyopadhyay, S., & Gelbukh, A. F. (2010). JU_CSE_TE: System description QA@ CLEF 2010-RespubliQA. In *CLEF (notebook papers/LABs/workshops).*

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning* (pp. 1310–1318).

Peñas, A., Forner, P., Rodrigo, Á., Sutcliffe, R. F. E., Forascu, C., et al. (2010). Overview of respubliqa 2010: Question answering evaluation over european legislation. In *CLEF 2010 LABs and workshops, notebook papers, 22-23 september 2010, Padua, Italy.*

Peñas, A., Forner, P., Sutcliffe, R., Rodrigo, Á., Forăscu, C., et al. (2010). Overview of respubliqa 2009: Question answering evaluation over european legislation. In *Multilingual information access evaluation i. text retrieval experiments* (pp. 174–196). Springer.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods*

in natural language processing (emnlp) (pp. 1532–1543).

Poibeau, T. (2003). *Extraction automatique d'information(du texte brut au web sémantique)*. Hermes science publications.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.

Qiu, X., Tian, L., & Huang, X. (2013). Latent semantic tensor indexing for community-based question answering. In *Acl (2)* (pp. 434–439).

Radev, D., Fan, W., Qi, H., Wu, H., & Grewal, A. (2005). Probabilistic question answering on the web. *Journal of the American Society for Information Science and Technology*, *56*(6), 571–583.

Riahi, F., Zolaktaf, Z., Shafiei, M., & Milios, E. (2012). Finding expert users in community question answering. In *Proceedings of the 21st international conference companion on World Wide Web* (pp. 791–798).

Robertson, S. E., & Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 232–241).

Robertson, S. E., Walker, S., Beaulieu, M., Gatford, M., & Payne, A. (1996). Okapi at TREC-4. In *Proceedings of the 4th text retrieval conference* (Vol. 500, pp. 73–97).

Rodrigo, Á., Perez-Iglesias, J., Peñas, A., Garrido, G., & Araujo, L. (2010). A question answering system based on information retrieval and validation. In *CLEF (notebook papers/LABs/workshops)*.

Romeo, S., Da San Martino, G., Barrón-Cedeno, A., Moschitti, A., Belinkov, Y., Hsu, W.-N., ... Glass, J. (2016). Neural attention for learning to rank questions in community question answering. In *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers* (pp. 1734–1745).

Romeo, S., Da San Martino, G., Belinkov, Y., Barrón-Cedeño, A., Eldesouki, M., Darwish, K., ... Moschitti, A. (2017). Language processing and learning models for community question answering in arabic. *Information Processing & Management*.

Rücklé, A., Swarnkar, K., & Gurevych, I. (2019). Improved cross-lingual question retrieval for community question answering. In *The world wide web conference* (pp. 3179–3186).

Sabnani, H., & Majumder, P. (2010). Question answering system: Retrieving relevant passages. In *CLEF (notebook papers/LABs/workshops)*.

Salton, G. (1968). *Automatic information organization and retrieval*. McGraw Hill Text.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text

retrieval. *Information Processing and Management*, *24*(5), 513–523.

Salton, G., & McGill, M. J. (1986). *Introduction to modern information retrieval*. New York, NY, USA: McGraw-Hill, Inc.

Saracevic, T. (1975). Relevance: A review of and a framework for the thinking on the notion in information science. *Journal of the American Society for Information Science*, *26*(6), 321–343.

Severyn, A., & Moschitti, A. (2012). Structural relationships for large-scale learning of answer re-ranking. In *Proceedings of the 35th international ACM SIGIR conference on research and development in information retrieval* (pp. 741–750).

Severyn, A., Nicosia, M., & Moschitti, A. (2013a). Building structures from classifiers for passage reranking. In *Proceedings of the 22nd ACM international conference on conference on information and knowledge management* (pp. 969–978).

Severyn, A., Nicosia, M., & Moschitti, A. (2013b). Learning adaptable patterns for passage reranking. In *Proceedings of the 7th conference on computational natural language learning* (pp. 75–83).

Shah, C., & Pomerantz, J. (2010). Evaluating and predicting answer quality in community qa. In *Proceedings of the 33rd international acm sigir conference on research and development in information retrieval* (pp. 411–418).

Shen, D., & Lapata, M. (2007). Using semantic roles to improve question answering. In *Proceedings of the empirical methods in natural language processing and computational natural language learning (EMNLP/CoNLL)* (pp. 12–21).

Singh, A. (2012). Entity based q&a retrieval. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning* (pp. 1266–1277).

Srikanth, M., & Srihari, R. (2002). Biterm language models for document retrieval. In *Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 425–426).

Sun, H., Ma, H., Yih, W.-t., Tsai, C.-T., Liu, J., & Chang, M.-W. (2015). Open domain question answering via semantic enrichment. In *Proceedings of the 24th international conference on World Wide Web* (pp. 1045–1055).

Suzuki, J., Sasaki, Y., & Maeda, E. (2002). SVM answer selection for open-domain question answering. In *Proceedings of the 19th international conference on computational linguistics-volume 1* (pp. 1–7).

Tamine-Lechani, L., & Calabretto, S. (2008). Recherche d'information contextuelle et web. *Recherche d'information: état des lieux et perspectives*, 201–224.

Tari, L., Tu, P. H., Lumpkin, B., Leaman, R., Gonzalez, G., & Baral, C. (2007). Passage relevancy through semantic relatedness. In *TREC*.

Tellex, S., Katz, B., Lin, J., Fernandes, A., & Marton, G. (2003). Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on research and development in informaion retrieval* (pp. 41–47).

Toba, H., Sari, S., Adriani, M., & Manurung, R. (2010a). Contextual approach for paragraph selection in question answering task. In *CLEF (notebook papers/LABs/workshops)*.

Toba, H., Sari, S., Adriani, M., & Manurung, R. (2010b). Contextual approach for paragraph selection in question answering task. In *CLEF (notebook papers/LABs/workshops)*.

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 384–394).

Verberne, S., Boves, L., Oostdijk, N., & Coppen, P.-A. (2008). Using syntactic information for improving why-question answering. In *Proceedings of the 22nd international conference on computational linguistics-volume 1* (pp. 953–960).

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3156–3164).

Voorhees, E. M. (2001). The TREC question answering track. *Journal of Natural Language Engineering*, *7*(04), 361–378.

Wang, K., Ming, Z., & Chua, T.-S. (2009). A syntactic tree matching approach to finding similar questions in community-based qa services. In *Proceedings of the 32nd international acm sigir conference on research and development in information retrieval* (pp. 187–194).

Widdows, D., & Ferraro, K. (2008). Semantic vectors: a scalable open source package and online technology management application. In *Lrec*.

Wilbur, W. J., & Sirotkin, K. (1992). The automatic identification of stop words. *Journal of Information Science*, *18*(1), 45–55.

Woods, W. A. (1973). Progress in natural language understanding: an application to lunar geology. In *Proceedings of the june 4-8, 1973, national computer conference and exposition* (pp. 441–450).

Wu, P.-C., Lee, Y.-S., & Yang, J.-C. (2008). Robust and efficient multiclass svm models for phrase pattern recognition. *Pattern recognition*, *41*(9), 2874–2889.

Wu, Z., & Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on association for computational linguistics* (pp. 133–138).

Xu, C., Bai, Y., Bian, J., Gao, B., Wang, G., Liu, X., & Liu, T.-Y. (2014). Rc-net: A general framework for incorporating knowledge into word representations.

In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management* (pp. 1219–1228).

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048–2057).

Xue, X., Jeon, J., & Croft, W. B. (2008). Retrieval models for question and answer archives. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval* (pp. 475–482).

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies* (pp. 1480–1489).

Ye, B., Feng, G., Cui, A., & Li, M. (2017). Learning question similarity with recurrent neural networks. In *2017 ieee international conference on big knowledge (icbk)* (pp. 111–118).

Yen, S.-J., Wu, Y.-C., Yang, J.-C., Lee, Y.-S., Lee, C.-J., & Liu, J.-J. (2013). A support vector machine-based context-ranking model for question answering. *Information Sciences*, *224*, 77–87.

Yu, M., & Dredze, M. (2014). Improving lexical embeddings with semantic knowledge. In *Acl (2)* (pp. 545–550).

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhai, C., & Lafferty, J. (2004). A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, *22*(2), 179–214.

Zhang, K., Wu, W., Wu, H., Li, Z., & Zhou, M. (2014). Question retrieval with high quality answers in community question answering. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management* (pp. 371–380).

Zhang, W.-N., Ming, Z.-Y., Zhang, Y., Liu, T., & Chua, T.-S. (2016). Capturing the semantics of key phrases using multiple languages for question retrieval. *IEEE Transactions on Knowledge and Data Engineering*, *28*(4), 888–900.

Zhou, G., Cai, L., Zhao, J., & Liu, K. (2011). Phrase-based translation model for question retrieval in community question answer archives. In *Proceedings of the 49th annual meeting of the acl: Human language technologies-volume 1* (pp. 653–662).

Zhou, G., He, T., Zhao, J., & Hu, P. (2015). Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of the 53rd annual meeting of the acl and the 7th international*

joint conference on natural language processing of the asian federation of natural language processing (pp. 250–259).

Zhou, G., Liu, Y., Liu, F., Zeng, D., & Zhao, J. (2013). Improving question retrieval in community question answering using world knowledge. In *Ijcai* (Vol. 13, pp. 2239–2245).

Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., & Xu, B. (2016). Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)* (pp. 207–212).